[project2]20191616

Created	@2023년 6월 4일 오후 8:21
Class	
Type	
Materials	
Reviewed	☐
파일과 미디어	

▼ BCNF 수정 및 physical schema 제작 과정

▼ Entity 변화

- service에 service_manager 추가
  - service_manager를 정하는 요소는 manger들의 일정, 업무량, 업무계획 등을 고려하여 정해지는 것이므로 프로젝트에 있는 데이터 요소에는 dependency가 없다. 따라서 BCNF에 영향을 미치지 않는다.
- track을 삭제
  - track의 starting_point와 destination은 service와 carry를 join해서 구하는 것이 더 정상적인 방법이다.
  - log라는 entity는 shipment_track로 개명하였다.
  - shipment_track은 service의 primary key인 service_ID와 shipment의 primary key인 shipment_ID를 foreign key로 가지고, 이 둘의 foreign key는 함께 shipment_track의 primary key가 된다.
- service에서 payment_type, payment_time, customer_ID_pay를 payment라는 entity로 따로 저장했다.
  - service의 정보를 담고 있는 엔티티에서 결제 정보까지 담을 경우, 속성이 다른 엔티티들과 비교했을 때 훨씬 많아진다.
  - 이로 인해 값의 중복이 많아진다. 중복이 많아지면 값의 일관성을 유지하기 힘들다.
  - payment_ID라는 속성을 추가하여 payment의 primary key로 지정한다.
  - payment_ID는 payment_type, payment_time, customer_ID를 결정한다. customer_ID는 customer_ID_pay의 이름을 바꾼 것이다. customer_ID가 하나이므로 service에서 했던 것처럼 foreign key의 이름을 변경할 필요가 없다.
  - payment_type, payment_time, customer_ID에서는 서로 간의 어떠한 종속성도 없으므로 BCNF를 위반하지 않는다.
  - payment에 pre_paid의 속성을 추가하여, 선불 후불 정보를 저장한다.
  - pre_paid는 payment에서 payment_ID를 제외한 어떠한 다른 속성에 종속되지 않는다.
  - service에서는 payment_ID를 속성으로 가지고 이는 payment에서 가져온 foreign_key이다.
  - 만약 service에서 payment_type, payment_time, customer_ID를 제거하지 않고 payment_ID와 pre_paid를 추가한다고 가정했을 때, payment_ID가 payment_type, payment_time, pre_paid, customer_ID를 결정하지만 payment_ID는 super_key가 아니므로 BCNF를 위반한다.
  - a→b가 BCNF를 위반할 때, R-b-a와 a+b로 분해하므로 payment_ID는 새로운 엔티티 payment에서 super_key이자 primary_key가 돼서 payment_type, payment_time, customer_ID, pre_paid를 결정한다.
  - payment_ID R(service)에서는 foreign key가 됨으로써 BCNF를 위배하지 않고 값을 저장한다.
- service에서 service_type과 cost를 분해했다.
  - sevice의 service_type은 cost를 결정한다. service_type에 따라 cost가 결정되므로 service_type → cost functional dependency는 BCNF를 위반한다. service_type이 super_key가 아니기 때문이다.
  - cost를 service entity에서 제외시키고 service_type과 service_cost를 따로 service_type_cost라는 entity에 저장하였다. service_type은 service에서 foreign key로 존재한다.

▼ Relation 변화

- service, package
  - service와 package는 일 대 일 관계이다. 하나의 service는 하나의 package만 관리하기로 결정했기 때문이다.
  - service를 가지지 않는 package가 있을 수 있다. service가 정해지지 않았을 수 있기 때문이다.

    - service는 package를 반드시 가져야한다. package가 없는 service는 무의미하기 때문이다.
- service, payment
    - service는 payment를 반드시 가진다. service에 대한 결제는 반드시 필요하기 때문이다.
    - payment 또한 어떠한 service는 반드시 하나만 가진다. 결제할 서비스 없이 결제내용이 존재하는 것은 말이 되지 않기 때문이다. payment는 하나의 service에 대한 결제내용만을 담고 있기 때문이다.
- customer, payment
    - customer는 여러 개의 payment를 가질 수 있다.
    - customer는 어떠한 payment도 가지지 않을 수 있다.
    - payment를 여러 명의 customer가 결제할 수 없다.
    - payment는 반드시 특정 한명의 customer에게 결제된다.
- service, customer
    - service는 반드시 sender와 reciever를 가진다.
    - sender와 reciever는 반드시 한 명이다.
    - 어떠한 customer는 여러 개의 service를 가질 수 있다. 동시에 어떠한 service도 갖지 않을 수도 있다. 이는 sender와 reciever에게 모두 해당된다.
- service, service_type_cost
    - service_type을 가지지 않는 service는 없다.
    - 하나의 service가 여러개의 service_type을 가질 수 없다.
    - 하나의 service_type에 해당하는 service가 여러 개일 수 있다.
    - 그러나 어떠한 service도 해당되지 않는 service_type이 존재할 수 있다.
- service, shipment_track
    - shipment_track은 service와 shipment의 관계를 many to many로 나타내기 위해 추가되었다.
    - 하나의 service는 여러 shipment로 구성될 수 있다.
    - shipment가 없는 service는 없다.
    - shipment는 하나 이상의 service에 반드시 속한다.
- shipment, shipment_service
    - shipment_track은 service와 shipment의 관계를 many to many로 나타내기 위해 추가되었다.
    - 하나의 shipment가 여러 service를 처리할 수 있다.
    - service가 없는 shipment 없다.
    - service는 하나 이상의 shipment는 무조건 가진다.
- shipment, transportation
    - shipment를 운영하는 transportation은 반드시 한개이며, 무조건 있어야 한다.
    - 어떠한 shipment를 진행 중이지 않은 transportation이 있을 수 있다.
    - 여러 shipment를 진행 중인 transportation이 있을 수 있다.
- shipment, location
    - shipment는 출발지와 도착지를 반드시 가진다.
    - 출발지와 도착지는 반드시 하나이다.
    - 여러 shipment가 특정 location을 출발지로 가질 수 있다.
    - 여러 shipment가 특정 lcoation을 도착지로 가질 수 있다.
- transportation, carry
    - carry는 package와 transportation의 many to many 관계를 나타내기 위함이다

    - many to many는 서로의 entity에서 primary_key를 foreign_key로 새로운 엔티티(shipment_track)로 가져온 후, 두 개의 foreign_key는 함께 새로운 엔티티(shipment_track)의 primary_key가 된다.
    - 하나의 package가 여러 transportation에 의해 carry 될 수 있다.
    - 하나의 transportation이 여러 package를 배송할 수 있다.
    - carry라 함은 반드시 transportation에 의해 이루어진다.
- carry, package
    - carry는 package와 transportation의 many to many 관계를 나타내기 위함이다
    - many to many는 서로의 entity에서 primary_key를 foreign_key로 새로운 엔티티(shipment_track)로 가져온 후, 두 개의 foreign_key는 함께 새로운 엔티티(shipment_track)의 primary_key가 된다.
    - 하나의 package가 여러 transportation에 의해 carry 될 수 있다.
    - 하나의 transportation이 여러 package를 배송할 수 있다.
    - carry는 반드시 package를 가진다.
- location, locates
    - locates는 location과 package의 many to many의 관계를 나타내기 위함이다.
    - many to many는 서로의 entity에서 primary_key를 foreign_key로 새로운 엔티티(shipment_track)로 가져온 후, 두 개의 foreign_key는 함께 새로운 엔티티(shipment_track)의 primary_key가 된다.
    - 하나의 package는 여러 location에 위치할 수 있다.
    - 하나의 location에 여러 package가 있을 수 있다.
    - locates는 반드시 location을 가져야 한다.
- package, locates
    - locates는 location과 package의 many to many의 관계를 나타내기 위함이다.
    - many to many는 서로의 entity에서 primary_key를 foreign_key로 새로운 엔티티(shipment_track)로 가져온 후, 두 개의 foreign_key는 함께 새로운 엔티티(shipment_track)의 primary_key가 된다.
    - 하나의 package는 여러 location에 위치할 수 있다.
    - 하나의 location에 여러 package가 있을 수 있다.
    - locates는 반드시 package를 가져야 한다.
- package, hazardous
    - hazardous는 반드시 하나의 package에 속한다.
    - package는 hazardous에서 foreign key이자 primary key이므로 identifying 관계이다.
    - 하나의 package가 여러 개의 hazardous를 가질 수 있다.
    - 어떠한 hazardous도 가지지 않은 package가 있을 수 있다.
- package, international
    - international은 반드시 하나의 package에 속한다.
    - package는 여러 개의 international를 가질 수 있다.
    - package는 어떠한 international도 가지지 않을 수 있다.
- carry와 locates모두 시간 순서대로 해당 package의 이동을 담고 있다.
    - carry는 운송수단에 따른 이동에 대한 기록을 담고 있는 반면, locates 위치하고 있던 장소에 따른 기록을 담고 있다.
    - 두 개의 entity 모두 package의 이동을 분석할 수 있기에 필요한 기준이 운송수단인지, 위치인지를 파악하여 기록에 대한 처리를 진행할 수 있다.

▼ Constraints

- service
    - service_ID
        - primary_key이므로 null이 허용되지 않는다.

        - service_ID의 자료형은 varchar(20)으로 하였다.
    - service_manager
        - null을 허용하지 않는다.
        - service_manager가 없을 경우, service를 관리할 사람이 없기 때문에 문제가 생긴다.
        - service_manger의 자료형은 varchar(20)으로 하였다.
    - service_date
        - null을 허용하지 않는다.
        - service가 시작되는 시간을 나타내기 때문에 null있으면 안된다.
        - service_date의 자료형은 timestamp으로 특정 시각을 나타낸다.
    - promised_time
        - null을 허용하지 않는다.
        - service가 할당되는 동시에 promised_time이 지정된다.
        - service_date와 마찬가지로 자료형은 timestamp로 특정 시각을 나타낸다.
    - status
        - null을 허용하지 않는다.
        - service의 상태를 나타내는 집합이 있다. 해당 집합에 해당하는 값만 가질 수 있다.
        - delivering, delivered, preparing으로 세 가지 값 중 하나를 가지며, 반드시 가져야한다.
        - 자료형은 varchar(20)이다.
    - payment_ID(FK)
        - null을 허용하지 않는다.
        - foreign_key이므로 null 값일 수 없다.
        - 결제되지 않는 service는 없다.
        - 무료 배송이라고 하더라도 payment는 반드시 존재한다.
        - 자료형은 varchar(20)이다.
    - customer_ID_sender(FK)
        - null을 허용하지 않는다.
        - foreign_key이므로 null 값일 수 없다.
        - 해당 package를 send하는 customer는 반드시 존재한다.
        - 자료형은 varchar(20)이다.
    - customer_ID_reciever(FK)
        - null을 허용하지 않는다.
        - foreign_key이므로 null 값일 수 없다.
        - 해당 package를 recieve하는 customer는 반드시 존재한다.
        - 자료형은 varchar(20)이다.
    - service_type(FK)
        - null을 허용하지 않는다.
        - foreign_key이므로 null 값일 수 없다.
        - 해당 service는 특정 type을 반드시 가진다.
        - 자료형은 varchar(20)이다.
    - package_ID(FK)
        - null을 허용하지 않는다.

        - foreign_key이므로 null 값일 수 없다.
        - 해당 service는 특정 package를 반드시 가진다.
- service_type_cost
    - service_type
        - null을 허용하지 않는다
        - primary_key이므로 null값을 가질 수 없다.
        - 자료형은 varchar(20)이다.
    - service_cost
        - null을 허용하지 않는다.
        - 해당 service의 type에 따른 가격을 나타낸다.
        - 자료형은 numeric이다.
- customer
    - customer_ID
        - null을 허용하지 않는다
        - primary_key이므로 null값을 가질 수 없다.
        - 자료형은 varchar(20)이다.
    - name
        - null을 허용하지 않는다
        - 이름이 없는 customer는 없다.
        - 자료형은 varchar(20)이다.
    - phone_number
        - null을 허용하지 않는다
        - 전화번호가 없는 customer는 없다. 반드시 연락처는 있어야 한다.
        - 자료형은 numeric이다.
    - account_number
        - null을 허용한다.
        - 신용카드로 결제하는 손님은 손님이 직접 결제를 진행하기 때문에 계좌번호가 필요없다.
        - 월 결제를 진행하는 손님의 계좌번호를 알고 있어야 자동결제를 진행할 수 있다.
        - 자료형은 numeric이다.
    - customer_address
        - null을 허용하지 않는다.
        - 주소를 알지 못한다면 배송상품을 배송할 수 없다.
        - 주소를 알지 못한다면 bill을 보낼 수도 없다.
        - 자료형은 varchar(20)이다.
- payment
    - payment_ID
        - null을 허용하지 않는다
        - primary_key이므로 null값을 가질 수 없다.
        - 자료형은 varchar(20)이다.
    - payment_type
        - null을 허용하지 않는다

        - payment_type은 두 가지 값 중 하나를 가진다.
        - monthly_bill 또는 credit_card이다.
        - 자료형은 varchar(20)이다.
    - payment_time
        - null을 허용한다.
        - 결제가 진행된 시간이다.
        - 아직 결제가 진행되지 않는 payment가 존재할 수 있다.
        - null은 아직 결제가 진행되지 않았음을 나타낸다.
        - 자료형은 timestamp로 특정시각을 나타낸다.
    - prepaid
        - payment_time이 null이라고해서 후불이나 선불을 판단할 수 없다.
        - 선불이지만 아직 결제가 진행되지 않은 것일 수도 있다.
        - 후불이지만 이미 결제가 진행된 것일 수도 있다.
        - prepaid는 0 또는 1의 값을 가진다.
        - 1은 선불, 0은 후불을 나타낸다.
        - 자료형은 numeric이다.
    - customer_ID(FK)
        - null을 허용하지 않는다.
        - foreign_key이므로 null 값일 수 없다.
        - 결제자가 누구인 지 저장하고 있다.
        - 결제자가 없는 payment는 있을 수 없다.
        - 자료형은 varchar(20)이다.
- shipment_track
    - shipment_ID(FK)
        - null을 허용하지 않는다.
        - foreign_key이므로 null 값일 수 없다.
        - shipment가 없는 service는 없다.
        - 자료형은 varchar(20)이다.
    - service_ID(FK)
        - null을 허용하지 않는다.
        - foreign_key이므로 null 값일 수 없다.
        - service가 없는 shipment는 없다.
        - 자료형은 varchar(20)이다.
- shipment
    - shipment_ID
        - null을 허용하지 않는다
        - primary_key이므로 null값을 가질 수 없다.
        - 자료형은 varchar(20)이다.
    - transportation_ID(FK)
        - null을 허용하지 않는다.
        - foreign_key이므로 null 값일 수 없다.

        - 운송수단이 없는 shipment는 없다.
        - 자료형은 varchar(20)이다.
    - location_ID_from(FK)
        - null을 허용하지 않는다.
        - foreign_key이므로 null 값일 수 없다.
        - 출발지가 없는 shipment는 없다.
        - 자료형은 varchar(20)이다.
    - location_ID_to(FK)
        - null을 허용하지 않는다.
        - foreign_key이므로 null 값일 수 없다.
        - 도착지가 없는 shipment는 없다.
        - 자료형은 varchar(20)이다.
- transportation
    - transportation_ID
        - null을 허용하지 않는다
        - primary_key이므로 null값을 가질 수 없다.
        - 자료형은 varchar(20)이다.
    - transportation_type
        - null을 허용하지 않는다.
        - airplane, ship, truck의 값 중 하나를 가진다.
        - 자료형은 varchar(20)이다.
    - current_location
        - null을 허용하지 않는다.
        - 해당 운송수단은 실물이기 때문에 반드시 현재 존재하는 공간이 있다.
        - 자료형은 varchar(20)이다.
- carry
    - transportation_ID(FK)
        - null을 허용하지 않는다.
        - foreign_key이므로 null 값일 수 없다.
        - 운송수단이 없는 carry는 없다.
        - 자료형은 varchar(20)이다.
    - pakcage_ID(FK)
        - null을 허용하지 않는다.
        - foreign_key이므로 null 값일 수 없다.
        - package가 없는 carry는 없다.
        - 자료형은 varchar(20)이다.
    - from_when
        - null을 허용하지 않는다.
        - carry가 할당되는 동시에 출발시간이 정해진다.
        - 출발 시각은 반드시 있다.
        - 자료형은 timestamp이다.

    - to_when
        - null을 허용한다.
        - 아직 도착하지 않았을 수 있다.
        - 여전히 이동 중인 경우 null값일 수 있다.
        - 자료형은 timestamp이다.
- location
    - location_ID
        - null을 허용하지 않는다
        - primary_key이므로 null값을 가질 수 없다.
        - 자료형은 varchar(20)이다.
    - location_type
        - null을 허용하지 않는다.
        - location_type은 반드시 존재한다.
        - location_type은 private, warehouse, port 중 하나다.
        - private은 사유지이다.
        - warehouse는 창고이다.
        - port는 배와 비행기가 도착하는 곳을 통틀어서 일컫는다.
        - 자료형은 varchar(20)이다.
    - location_address
        - null을 허용하지 않는다.
        - location의 주소는 반드시 존재한다.
        - 자료형은 varchar(20)이다.
- locates
    - location_ID(FK)
        - null을 허용하지 않는다.
        - foreign_key이므로 null 값일 수 없다.
        - location이 없는 locates는 존재할 수 없다.
        - 자료형은 varchar(20)이다.
    - package_ID(FK)
        - null을 허용하지 않는다.
        - foreign_key이므로 null 값일 수 없다.
        - package가 없는 locates는 존재할 수 없다.
        - 자료형은 varchar(20)이다.
    - from_when
        - null을 허용하지 않는다.
        - locates가 할당되는 동시에 시작시간이 정해진다.
        - 자료형은 timestamp이다.
    - to_when
        - null을 허용한다.
        - 현재 여전히 해당 location에 package가 존재할 수 있다.
        - 이와 같은 경우 값을 null로 하여 아직도 해당 location에 package가 있음을 나타낸다.

        - 자료형은 timestamp이다.
- package
    - package_ID
        - null을 허용하지 않는다
        - primary_key이므로 null값을 가질 수 없다.
        - 자료형은 varchar(20)이다.
    - weight
        - null을 허용하지 않는다.
        - package의 무게는 반드시 존재한다.
        - 자료형은 numeric이다.
    - package_type
        - null을 허용하지 않는다.
        - 해당 회사에서 처리하는 package의 종류는 정해져 있고, 해당 종류 중 하나의 값을 반드시 가진다.
        - package의 종류는 'document', 'box', 'envelope', 'parcel'가 있다.
        - 자료형은 varchar(20)이다.
    - timeline
        - null을 허용하지 않는다.
        - package의 일정이다.
        - package의 일정 종류는 정해져있고, 해당 값 중 반드시 하나를 가진다.
        - package의 일정 종류는 'express', 'standard', 'overnight', 'weekend'가 있다.
        - 자료형은 varchar(20)이다.
- hazardous
    - package_ID(FK)
        - null을 허용하지 않는다.
        - foreign_key이므로 null 값일 수 없다.
        - 자료형은 varchar(20)이다.
    - content
        - null을 허용하지 않는다.
        - primary_key이므로 null 값일 수 없다.
        - 자료형은 varchar(20)이다.
    - type
        - null을 허용하지 않는다.
        - 어떠한 위험종류인지는 반드시 나타낸다.
        - 위험종류 중 반드시 하나의 값을 가진다.
        - 위험종류는 'flammable', 'toxic', 'explosive'와 같다.
        - 자료형은 varchar(20)이다.
- international
    - declaration_ID
        - null을 허용하지 않는다.
        - primary_key이므로 null 값일 수 없다.
        - 자료형은 varchar(20)이다.

- content
  - null을 허용하지 않는다.
  - 어떠한 내용물인지는 반드시 나타낸다.
  - 자료형은 varchar(20)이다.
- price
  - null을 허용하지 않는다.
  - 국제배송물건이므로 얼마인지는 반드시 나타낸다.
  - 자료형은 numeric(20)이다.
- package_ID(FK)
  - null을 허용하지 않는다.
  - foreign_key null값일 수 없다.
  - 어떠한 package에 속해있는 지 나타낸다.
  - 자료형은 varchar(20)이다.

▼ Database 제작과정

▼ DDL(Data Description Language)

- DDL.txt파일을 통해 table을 생성한다.
- DDL.txt에는 create table query를 사용하여 table들을 제작한다.
- create table service
  (
  service_ID varchar(20),
  service_date varchar(20),
  service_type varchar(20),
  check (service_type in ('level_1', 'level_2', 'level_3')),
  status varchar(20),
  check (status in ('delivered', 'delivering', 'preparing')),
  service_manager varchar(20),
  payment_ID varchar(20),
  customer_ID_sender varchar(20),
  customer_ID_reciever varchar(20),
  package_ID varchar(20),
  primary key (service_ID),
  foreign key (payment_ID) references payment (payment_ID)
  on delete cascade,
  foreign key (customer_ID_sender) references customer(customer_ID)
  on delete cascade,
  foreign key (customer_ID_reciever) references customer(customer_ID)
  on delete cascade,
- foreign key (service_type) references service_type_cost(service_type),
  foreign key (package_ID) references package(package_ID)
  on delete cascade
  );
- 위는 service의 table을 제작한 예시이다.
- 각 attribute의 이름과 해당 attribute의 자료형을 나열한다.
- check query를 통해 constraints를 검사한다.
- 어떤 attribute이 primary key인 지를 나타낸다.
- 어떤 attribute이 foreign key인 지를 나타낸다.
- 어떤 table에서부터 온 foreign key인지도 나타낸다.

- foreign key를 나타낼 때, 해당 table의 data와의 동기화는 어떻게 처리할 것인 지 정한다.
- table을 실행할 때는 foreign_key가 참조하는 table이 이미 create되어있어야 한다.
- 이미 create되어 있지 않은 table을 참조해서 table을 생성하려는 경우 에러가 발생한다.

▼ DDL+drop(Data Description Language)

- DDL+drop은 databse를 update하거나 처음부터 다시 만들기 위해 사용된다.
- DDL+drop은 drop기능이 추가된 것으로 이미 만들어져 있는 table과 table에 해당하는 data들을 모두 삭제하고, 다시 create table query를 실행한다.
- 특정 table을 drop할 때는 해당 table을 참조하고 있는 table부터 먼저 drop해야한다.
- 그렇지 않고 참조되고 있는 table을 먼저 drop한다면 해당 table을 참조하고 있는 table을 drop할 때 에러가 발생한다.
- drop table shipment_track;
  drop table shipment;
  drop table carry;
  drop table locates;
  drop table transportation;
  drop table location;
- 위는 drop의 예시이다.
- shipment_track은 shipment의 shipment_ID를 참조하고 있다.
- 그렇기 때문에 shipment_track을 먼저 drop하고 shipment를 drop해야한다.
- 마찬가지로, carry는 transportation의 transportation_ID를 참조하고 있고, locates는 location의 location_ID를 참조하고 있다.
- 따라서, transportation을 drop하기 이전에 carrys를 drop해야하고, location을 drop하기 전에 locates를 drop해야한다.

▼ InsertFile

- 생성된 table에 실제 data를 넣는 query들을 저장하고 있는 file이다.
- table에 data를 넣을 대는 insert query를 사용한다.
- insert into payment values ('payment_b_1', 'credit_card', '2023-05-16 09:00:00', '1','customer_b');
  insert into service values ('c_to_b', '2023-05-17 09:00:00', 'level_3', 'delivered', 'kun', 'payment_b_1', 'customer_c', 'customer_b','package_b');
  insert into locates values ('port_a', 'package_b', '2023-05-17 09:00:00', '2023-05-17 10:00:00');
  insert into carry values ('airplane_a', 'package_b', '2023-05-17 10:00:00', '2023-05-17 15:00:00');
  insert into locates values ('warehouse_b', 'package_b', '2023-05-17 15:00:00', '2023-05-17 17:00:00');
  insert into carry values ('ship_a', 'package_b', '2023-05-17 17:00:00', '2023-05-17 18:00:00');
  insert into locates values ('warehouse_c', 'package_b', '2023-05-17 18:00:00', '2023-05-17 19:00:00');
  insert into carry values ('truck_b', 'package_b', '2023-05-17 19:00:00', '2023-05-17 21:00:00');
  insert into locates values ('private_b', 'package_b', '2023-05-17 21:00:00', '2023-05-17 21:00:00');
  insert into shipment values ('package_b_1', 'airplane_a', 'port_a', 'warehouse_b');
  insert into shipment values ('package_b_2', 'ship_a', 'warehouse_b', 'warehouse_c');
  insert into shipment values ('package_b_3', 'truck_b', 'warehouse_c', 'private_b');
  insert into shipment_track values ('c_to_b', 'package_b_1');
  insert into shipment_track values ('c_to_b', 'package_b_2');
  insert into shipment_track values ('c_to_b', 'package_b_3');
- 위는 insert의 예시이다.
- 각 table이 foreign_key로 물려있기 때문에 table별로 값이 일치해야 join을 할 때 문제가 없다.
- locates carry가 번갈아 insert되면서 실제 배송을 상상할 수 있다.
- locates의 종료시간과 carry의 출발시간이 현실적으로 정확하게 일치할 수는 없지만 시간의 역순은 발생하면 안되는 것을 고려하여 값을 설정한다..
- carry와 locates모두 from_when값이 to_when보다 늦을 수 없는 것을 고려하여 값을 설정한다.
- shipment는 locates, carrys, locates에 해당하는 location, transportation, location이 저장됨에 따라 테이블을 연결해주는 것을 확인할 수 있다.

- 그리고 해당 shipment들은 shipment_track에 저장되어 해당 service의 track으로 확인할 수 있다.

▼ Code Implementation

전체적인 logic은 다음과 같다. type에 따라 찾고 싶은 내용이 있다. type에 따라 찾고 싶은 구체적인 내용을 input으로 입력받는다 해당 내용을 찾을 수 있게 query를 만든 후, database에 보내면 찾은 query를 통해 받은 결과값들을 받을 수 있다. 받은 결과값을 형식에 맞게 출력한다.

▼ 코드 시작 부분

```
#include <stdio.h>
#include "mysql.h"
```

- printf나 scanf같은 표준 라이브러리 함수를 사용하기 위해 stdio 라이브러리를 include 한다.
- MySQL관련 함수들을 사용하기 위해 mysql.h를 include해야한다. mysql.h를 include해야 이후에 사용할 MySQL함수를 문제 없이 사용할 수 있다.

```
#pragma comment(lib, "libmysql.lib")
#pragma warning(disable:4996)
```

- MySQL C API를 사용하기 위해 libmysql.lib 라이브러리를 링크한다.
- 4996번 에러를 무시하기 위해 사용한다. scanf할 때 보안문제를 통과하기 위함이다.

```
#define MAX_QUERY 10000
const char* host = "localhost";
const char* user = "root";
const char* pw = "Dlrjsghk12@";
const char* db = "delivery";
```

- 연결할 Database의 정보를 저장한다.
- query의 가장 긴 길이는 10000이다.

▼ 데이터 베이스와 연결하기

```
int main(void) {

  MYSQL* connection = NULL;
  MYSQL conn;
  MYSQL_RES* sql_result;
  MYSQL_ROW sql_row;
```

- 연결정보를 저장하고 query를 통해 구한 결과를 저장할 변수들이다.

```
if (mysql_init(&conn) == NULL)
    printf("mysql_init() error!");
```

- MySQL연결을 위해 초기화한다.

```
connection = mysql_real_connect(&conn, host, user, pw, db, 3306, (const char*)NULL, 0);
  if (connection == NULL)
  {
    printf("%d ERROR : %s\n", mysql_errno(&conn), mysql_error(&conn));
    return 1;
  }
```

- MySQL을 연결하고 해당 연결이 실패하면 에러메시지를 출력한다.

▼ 데이터베이스 스키마 선택하기

```
printf("Connection Succeed\n");

    if (mysql_select_db(&conn, db))
```

```
    {
      printf("%d ERROR : %s\n", mysql_errno(&conn), mysql_error(&conn));
      return 1;
    }
```

- 연결을 완료한 후, 선택한 db와의 연결을 시도한다. 실패하면 에러메시지를 출력한다.

▼ 데이터베이스 생성하기

```
char query[MAX_QUERY] = "";
    char line[MAX_QUERY] = "";
    int drop_need = 0;

    FILE* ddlFile = fopen("DDL.txt", "r");
    if (ddlFile == NULL)
    {
      printf("Failed to open DDL.txt\n");
      return 1;
    }
    while (fgets(line, sizeof(line), ddlFile) != NULL)
    {
      strcat(query, line);

      // Execute query if it ends with a semicolon
      if (strlen(query) > 0 && query[strlen(query) - 2] == ';')
      {
        if (mysql_query(connection, query) != 0)
        {
          drop_need = 1;
          break;
        }
        query[0] = '\0'; // Reset query
      }
    }
    fclose(ddlFile);
```

- ddl.txt 을 통해 데이터베이스를 생성한다.
- ddl.txt는 데이터베이스에 table이 없을 때만 생성가능하다.
- query는 MAX_QUERY의 충분한 공간을 가지고 있다. QUERY는 한 줄씩 입력받아 하나의 query가 완성되면 database에 해당 query를 전송한다.
- state는 반환값으로 query가 성공적으로 작동했을 때 0을 return한다.
- 그러나 만약 이미 table이 있어서 생성이 불가능할 경우 drop이 필요하다는 뜻이다.
- query전송 후 실패하면 반복문을 탈출하고 drop이 필요하다는 뜻인 drop_need의 값을 1로 설정한다.
- 해당 flag를 1로 설정하여 다음 code에서 drop을 먼저하고 다시 table을 최신화 할 수 있도록 한다.

```
if (drop_need)
    {
      FILE* ddldropFile = fopen("DDLdrop.txt", "r");
      if (ddldropFile == NULL)
      {
        printf("Failed to open DDLdrop.txt\n");
        return 1;
      }

      char query[MAX_QUERY] = "";
      char line[MAX_QUERY] = "";
      while (fgets(line, sizeof(line), ddldropFile) != NULL)
      {
        strcat(query, line);

        // Execute query if it ends with a semicolon
        if (strlen(query) > 0 && query[strlen(query) - 2] == ';')
        {
          if (mysql_query(connection, query) != 0)
          {
            printf("DDLdrop %d ERROR : %s\n", mysql_errno(connection), mysql_error(connection));
            break;
          }
          query[0] = '\0'; // Reset query
        }
      }
      fclose(ddldropFile);
    }
```

- drop_need가 1이라면 drop을 먼저 해야하는 것이므로 drop이 먼저 실행되고 create이 실행되는 DDLdrop.txt파일을 실행한다.
- 한줄씩 읽으면서 하나의 query가 완성되면 해당 query를 데이터베이스에 보낸다.
- state는 반환값으로 query가 성공적으로 작동했을 때 0을 return한다.
- query전송후 실패하면 반복문을 탈출한다.

```
FILE* insertFile = fopen("Insert.txt", "r");
    if (insertFile == NULL)
    {
      printf("Failed to open Insert.txt\n");
      return 1;
    }

    while (fgets(line, sizeof(line), insertFile) != NULL)
    {
      strcat(query, line);

      // Execute query if it ends with a semicolon
      if (strlen(query) > 0 && query[strlen(query) - 2] == ';')
      {
        if (mysql_query(connection, query) != 0)
        {
          printf("Insert %d ERROR : %s\n", mysql_errno(connection), mysql_error(connection));
          break;
        }
        query[0] = '\0'; // Reset query
      }
    }
    fclose(insertFile);
```

- table이 완성되었으므로 insert.txt파일에서 한줄씩 입력받아 query를 만든다.
- 하나의 query가 완성되면 해당 query를 데이터베이스에 전송한다.
- state는 반환값으로 query가 성공적으로 작동했을 때 0을 return한다.
- query전송 후 실패하면 반복문을 탈출한다.

▼ TYPE 입력받기

```
    int type = 0;
    char query[MAX_QUERY];
    do
    {
      printf("------- SELECT QUERY TYPES -------\n\n");
      printf("\t1. TYPE 1\n");
      printf("\t2. TYPE 2\n");
      printf("\t3. TYPE 3\n");
      printf("\t4. TYPE 4\n");
      printf("\t5. TYPE 5\n");
      printf("\t0. QUIT\n");
      int state = 0;
      int temp_ret = scanf("%d", &type);
      if (type == 1){...}
      else if (type == 2) {...}
      else if (type == 3) {...}
      else if (type == 4) {...}
      else if (type == 5) {...}
    } while (type != 0);

    mysql_close(connection);
```

- 기본적인 로직은 위와 같다. type을 보기로 제공해준 후, type에 맞는 조건문을 실행시키는 간단한 구조이다. 0이 입력될 때까지 계속 run해야하므로 do-while문을 사용하였다.

▼ TYPE 1

```
do
{
  printf("------- Subtypes in TYPE 1 -------\n");
  printf("\t1. TYPE 1-1\n");
  printf("\t2. TYPE 1-2\n");
  printf("\t3. TYPE 1-3\n");
  int temp_ret = scanf("%d", &subtype);
```

```
  if (subtype == 1){...}
  else if (subtype == 2){...}
  else if (subtype == 3){...}
} while(subtype != 0);
```

- 이전에 나온 코드와 마찬가지로 비슷한 로직을 나타내고 있다. subtype으로 0이 입력될 때까지 do-while문 속에서 계속 run되면서 subtype에 따라 코드가 실행된다.

▼ Subtype 1

```
if (subtype == 1)
          {
            char when[100];  // 2023-05-17 12:00:00
            char truck[100]; // airplane_b
            char temp;
            printf("---- TYPE 1-1 ----\n");
            printf("**Find all customers who had a package on the truck at the time of the crash.**\n");
```

- 1인 subtype을 골랐을 경우 truck_ID를 input으로 입력받는다.
- truck의 사고 시점을 알아야 해당 사고 상황을 정확히 파악하여 문제에 답할 수 있다.

```
            while ((temp = getchar()) != EOF)
            {
              if (temp == '\n' && i != 0)
                break;
              if (temp != '\n')
                truck[i++] = temp;
            }
            truck[i] = '\0';
            printf("When? : ");
            i = 0;
            while ((temp = getchar()) != EOF)
            {
              if (temp == '\n' && i != 0)
                break;
              if (temp != '\n')
                when[i++] = temp;
            }
            when[i] = '\0';
            sprintf(query, "select distinct customer_ID_sender from carry natural join service where transportation_ID = '%

            state = mysql_query(connection, query);
```

- 위의 코드는 transportation_ID와 사고시각을 입력받는 코드이다.
- 입력받은 값을 query에 각자 해당하는 자리에 삽입하여 database에 전송한다.
- state는 반환값으로 query가 성공적으로 작동했을 때 0을 return한다.

```
if (state == 0)
            {
              printf("[RESULT : (Customer_ID)Sender who had a package on that truck]\n");
              sql_result = mysql_store_result(connection);
              unsigned int num_rows = mysql_num_rows(sql_result);
              while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
              {
                printf("%s ", sql_row[0]);
                printf("\n");
              }
              mysql_free_result(sql_result);
```

- sql_result의 row개수가 0이라는 뜻은 없다는 뜻이므로 [NONE]을 출력한다.
- sql_result에 저장된 결과값을 sql_row로 하나씩 받아서 출력한다. 위의 경우, 결과값 속성 값이 하나밖에 없으므로 sql_row[0]만 출력한다.

▼ Subtype 2

```
else if (subtype == 2)
          {
            char when[100];  // 2023-05-17 12:00:00
            char truck[100]; // airplane_b
```

```
            char temp;
            printf("---- TYPE 1-2 ----\n");
            printf("**Find all recipients who had a package on that truck at the time of the crash.**\n");
            printf("Which Truck? : ");
            int i = 0;
            while ((temp = getchar()) != EOF)
            {
              if (temp == '\n' && i != 0)
                break;
              if (temp != '\n')
                truck[i++] = temp;
            }
            truck[i] = '\0';
            printf("When? : ");
            i = 0;
            while ((temp = getchar()) != EOF)
            {
              if (temp == '\n' && i != 0)
                break;
              if (temp != '\n')
                when[i++] = temp;
            }
            when[i] = '\0';
            sprintf(query, "select distinct customer_ID_reciever from carry natural join service where transportation_ID =

            state = mysql_query(connection, query);
```

- subtype 2의 경우도 subtype 1과 유사하다. 차이는 query내용밖에 없다.
- subtype 1과 마찬가지로, query에 필요한 입력값을 받아 query에 각자 해당하는 자리에 넣어준 후, 해당 query를 데이터베이스에 보낸다.
- state가 0일 경우 query가 정상작동했다는 뜻이며, 해당 query의 결과값을 아래의 토드를 사용하여 출력한다.

```
if (state == 0)
            {
              printf("[RESULT : (Customer_ID)Recipients who had package on that truck]\n");
              sql_result = mysql_store_result(connection);
              unsigned int num_rows = mysql_num_rows(sql_result);
              if (num_rows == 0)
                printf("[NONE]\n");
              while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
              {
                printf("%s ", sql_row[0]);
                printf("\n");
              }
              mysql_free_result(sql_result);
            }
```

- sql_result를 통해 결과값을 받고, 결과 값을 한줄씩 출력한다.
- 결과값의 속성이 한 개밖에 없으므로 sql_row[0]만 출력해도 무방한다.
- sql_result의 row개수가 0이라는 뜻은 없다는 뜻이므로 [NONE]을 출력한다.

▼ Subtype 3

```
char when[100];  // 2023-05-17 12:00:00
            char truck[100]; // airplane_b
            char temp;
            printf("---- TYPE 1-3 ----\n");
            printf("**Find the last successful delivery by that truck prior to the crash.**\n");
            printf("Which Truck? : ");
            int i = 0;
            while ((temp = getchar()) != EOF)
            {
              if (temp == '\n' && i != 0)
                break;
              if (temp != '\n')
                truck[i++] = temp;
            }
            truck[i] = '\0';
            printf("When? : ");
            i = 0;
            while ((temp = getchar()) != EOF)
            {
              if (temp == '\n' && i != 0)
                break;
              if (temp != '\n')
                when[i++] = temp;
```

```
            }
            when[i] = '\0';
            sprintf(query, "select distinct location_ID_from from carry natural join shipment where transportation_ID = '%s

            state = mysql_query(connection, query);
```

- subtype 1, 2와 마찬가지로 query에 필요한 입력값을 받는다.
- 입력받은 값을 query에 넣어준 후, 해당 query를 database에 보낸다.
- query가 정상적으로 작동했을 경우, state이 0이다.
- state이 0이라면 결과값을 아래와 같이 출력한다.

```
if (state == 0)
            {
              printf("[RESULT : (location_ID)Last successful delivery location]\n");
              sql_result = mysql_store_result(connection);
              unsigned int num_rows = mysql_num_rows(sql_result);
              if (num_rows == 0)
                printf("[NONE]\n");
              while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
              {
                printf("%s ", sql_row[0]);
                printf("\n");
              }
              mysql_free_result(sql_result);
            }
```

- sql_result의 row개수가 0이라는 뜻은 없다는 뜻이므로 [NONE]을 출력한다.
- sql_result에 저장된 결과값을 sql_row로 하나씩 받아서 출력한다. 위의 경우, 결과값 속성 값이 하나밖에 없으므로 sql_row[0]만 출력한다.

```
Connection Succeed
------- SELECT QUERY TYPES -------

        1. TYPE 1
        2. TYPE 2
        3. TYPE 3
        4. TYPE 4
        5. TYPE 5
        0. QUIT
1
------- Subtypes in TYPE 1 -------
        1. TYPE 1-1
        2. TYPE 1-2
        3. TYPE 1-3
1
---- TYPE 1-1 ----
**Find all customers who had a package on the truck at the time of the crash.**
Which Truck? : truck_b
When? : 2023-05-17 20:00:00
[RESULT : (Customer_ID)Sender who had a package on that truck]
customer_c
------- Subtypes in TYPE 1 -------
        1. TYPE 1-1
        2. TYPE 1-2
        3. TYPE 1-3
2
---- TYPE 1-2 ----
**Find all recipients who had a package on that truck at the time of the crash.**
Which Truck? : truck_b
When? : 2023-05-17 20:00:00
[RESULT : (Customer_ID)Recipients who had package on that truck]
customer_b
------- Subtypes in TYPE 1 -------
        1. TYPE 1-1
        2. TYPE 1-2
        3. TYPE 1-3
3
---- TYPE 1-3 ----
**Find the last successful delivery by that truck prior to the crash.**
Which Truck? : truck_b
When? : 2023-05-17 20:00:00
[RESULT : (location_ID)Last successful delivery location]
warehouse_c
------- Subtypes in TYPE 1 -------
        1. TYPE 1-1
        2. TYPE 1-2
        3. TYPE 1-3
```

▼ TYPE 2

- type 1에서 0을 입력하자 원래 type 선택문을 볼 수 있었다.
- Type 2같은 경우 query의 종류가 하나이고, 몇년도인지만 입력받으면 된다.

```
char when[100];
        char temp;
        int i;
        printf("------- TYPE 2 -------\n");
        printf("**Find the customer who has shipped the most packages in the past year**\n");
        printf("Which year? : ");
        i = 0;
        while ((temp = getchar()) != EOF)
        {
          if (temp == '\n' && i != 0)
            break;
          if (temp != '\n')
            when[i++] = temp;
        }
        int current_year = atoi(when);
        int next_year = atoi(when) + 1;
        sprintf(query, "SELECT c.customer_ID, c.name, COUNT(*) AS send_count FROM service s JOIN customer c ON s.customer_ID_se
        state = mysql_query(connection, query);
```

- 위의 코드와 같이 when이라는 문자열에 입력값을 받는다.
- 년도는 숫자이므로 현재년도와 다음년도의 값을 int형으로 하여 저장할 수 있게 한다.
- query에 넣을 때 query가 잘 작동할 수 있도록 앞을 0으로 채워 네 자리수가 될 수 있게 넣는다.
- 입력받은 값을 query에 넣어준 후, 해당 query를 database에 보낸다.
- query가 정상적으로 작동했을 경우, state이 0이다.
- state이 0이라면 결과값을 아래와 같이 출력한다.

```
if (state == 0)
        {
          printf("[RESULT : (customer_ID, name)Customer who has spent the most money]\n");
          sql_result = mysql_store_result(connection);
          unsigned int num_rows = mysql_num_rows(sql_result);
          if (num_rows == 0)
            printf("[NONE]\n");
          while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
          {
            printf("%s ", sql_row[0]);
            printf("%s", sql_row[1]);
            printf("\n");
          }
          mysql_free_result(sql_result);
        }
```

- sql_result의 row개수가 0이라는 뜻은 없다는 뜻이므로 [NONE]을 출력한다.
- sql_result에 저장된 결과값을 sql_row로 하나씩 받아서 출력한다. 위의 경우, 결과값 속성 값이 하나밖에 없으므로 sql_row[0]만 출력한다.

```
------- Subtypes in TYPE 1 -------
        1. TYPE 1-1
        2. TYPE 1-2
        3. TYPE 1-3
0
------- SELECT QUERY TYPES -------

        1. TYPE 1
        2. TYPE 2
        3. TYPE 3
        4. TYPE 4
        5. TYPE 5
        0. QUIT
2
------- TYPE 2 -------
**Find the customer who has shipped the most packages in the past year**
Which year? : 2023
[RESULT : (customer_ID, name)Customer who has shipped most]
customer_c name_c
```

▼ TYPE 3

```
else if (type == 3)
      {
        printf("------- TYPE 3 -------\n");
        printf("**Find the customer who has spent the most money on shipping in the past year.**\n");
        char when[100];
        char temp;
        int i;
        printf("Which year? : ");
        i = 0;
        while ((temp = getchar()) != EOF)
        {
          if (temp == '\n' && i != 0)
            break;
          if (temp != '\n')
            when[i++] = temp;
        }
        when[i] = '\0';
        int current_year = atoi(when);
        int next_year = atoi(when) + 1;
```

```
        sprintf(query, "SELECT c.customer_ID, c.name FROM service s JOIN payment p ON s.payment_ID = p.payment_ID JOIN customer
        state = mysql_query(connection, query);
```

- 위의 코드와 같이 when이라는 문자열에 입력값을 받는다.
- 년도는 숫자이므로 현재년도와 다음년도의 값을 int형으로 하여 저장할 수 있게 한다.
- query에 넣을 때 query가 잘 작동할 수 있도록 앞을 0으로 채워 네 자리수가 될 수 있게 넣는다.
- 입력받은 값을 query에 넣어준 후, 해당 query를 database에 보낸다.
- query가 정상적으로 작동했을 경우, state이 0이다.
- state이 0이라면 결과값을 아래와 같이 출력한다.

```
if (state == 0)
        {
          printf("[RESULT : (customer_ID, name)Customer who has spent the most money]\n");
          sql_result = mysql_store_result(connection);
          unsigned int num_rows = mysql_num_rows(sql_result);
          if (num_rows == 0)
            printf("[NONE]\n");
          while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
          {
            printf("%s ", sql_row[0]);
            printf("%s", sql_row[1]);
            printf("\n");
          }
          mysql_free_result(sql_result);
        }
```

- sql_result의 row개수가 0이라는 뜻은 없다는 뜻이므로 [NONE]을 출력한다.
- sql_result에 저장된 결과값을 sql_row로 하나씩 받아서 출력한다. 위의 경우, 결과값 속성 값이 하나밖에 없으므로 sql_row[0]만 출력한다.

```
3
------- TYPE 3 -------
**Find the customer who has spent the most money on shipping in the past year.**
Which year? : 2023
[RESULT : (customer_ID, name)Customer who has spent the most money]
customer_a name_a
------- SELECT QUERY TYPES -------

        1. TYPE 1
        2. TYPE 2
        3. TYPE 3
        4. TYPE 4
        5. TYPE 5
        0. QUIT
```

▼ TYPE 4

```
else if (type == 4)
      {
        printf("------- TYPE 4 -------\n");
        printf("**Find those packages that were not delivered within the promised time.**\n");
        query[0] = '\0';
        sprintf(query, "SELECT p.package_ID FROM package p JOIN service s ON p.package_ID = s.package_ID JOIN(SELECT package_ID
        state = mysql_query(connection, query);
```

- type 4의 경우 입력받아야 하는 값이 필요 없다.
- 만들어 놓은 해당 query를 database에 보낸다.
- query가 정상적으로 작동했을 경우, state이 0이다.
- state이 0이라면 결과값을 아래와 같이 출력한다.

```
4
------- TYPE 4 -------
**Find those packages that were not delivered within the promised time.**
[RESULT : (package_ID)pacakages not delivered within the promised time]
package_b
package_f
package_h
------- SELECT QUERY TYPES -------

        1. TYPE 1
        2. TYPE 2
        3. TYPE 3
        4. TYPE 4
        5. TYPE 5
        0. QUIT
```

▼ TYPE 5

```
if (type == 5)
      {
        char year[100];
        char month[100];
        int i;
        char temp;
        printf("------- TYPE 5 -------\n");
        printf("**Generate the bill for each customer for the past month. Consider creating several types of bills.**\n");
        printf("Which year? : ");
        i = 0;
        while ((temp = getchar()) != EOF)
        {
          if (temp == '\n' && i != 0)
            break;
          if (temp != '\n')
            year[i++] = temp;
        }
        year[i] = '\0';
        printf("Which month? : ");
        i = 0;
        while ((temp = getchar()) != EOF)
        {
          if (temp == '\n' && i != 0)
            break;
          if (temp != '\n')
            month[i++] = temp;
        }
        month[i] = '\0';
        int cur_year = atoi(year);
        int cur_month = atoi(month);
        int next_month = cur_month + 1;
```

- 연도와 달을 입력으로 받는다. year와 month 모두 숫자이므로 입력받은 후, int형으로 저장한다.
- bill의 type별로 출력해야하는데, bill의 type은 2가지이고, 둘의 출력 형식은 유사하다.
- 따라서 아래와 같이 for문으로 처리하였다.

```
for (int i = 1; i < 3; i++)
        {
          bill = i;
          if (bill == 1)
            strcpy(bill_type, "monthly_pay");
          else if (bill == 2)
            strcpy(bill_type, "credit_card");
          sprintf(query, "SELECT c.*, p.*, s.* FROM payment p NATURAL JOIN service s NATURAL JOIN customer c WHERE p.payment_ty
          state = mysql_query(connection, query);
```

- bill이 1일 경우, monthly_pay이 bill type이며, 2일 경우 credit_card가 bill_type이다.
- 입력받은 값과 bill_type을 함께 query에 각자 해당하는 자리에 넣어서 데이터베이스로 보낸다. 즉, TYPE5에서는 query가 두 번 보내지는 것이다.
- state가 0일 경우, query가 정상작동했다는 뜻이며 해당 query의 결과값을 아래의 코드를 사용하여 형식에 맞게 출력한다.

```
if (state == 0)
          {
            printf("<bill type : %s>\n", bill_type);
            sql_result = mysql_store_result(connection);
            unsigned int num_rows = mysql_num_rows(sql_result);
            if (num_rows == 0)
              printf("[NONE]\n");
            char current_customer[100] = "";

            while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
            {
              if (strcmp(current_customer, sql_row[0]) != 0)
              {
                if (current_customer[0] != '\0')
                  printf("\n");
                for (unsigned int i = 0; i < 5; i++)
                {
                  if (i == 0)
                    printf("customer_ID : %-10s\n", sql_row[i]);
                  if (i == 1)
                    printf("name : %-10s\n", sql_row[i]);
                  if (i == 2 && bill == 1)
                    printf("account : %-10s\n", sql_row[i]);
                  if (i == 3)
                    printf("phone number : %-10s\n", sql_row[i]);
                  if (i == 4)
                    printf("address : %-10s\n", sql_row[i]);
                }
                printf("[payment_ID]   [payment_time]      [prepaid?]   [service_ID]  [type]        [status]      [manager]
                strcpy(current_customer, sql_row[0]);
              }
              for (unsigned int i = 5; i < mysql_num_fields(sql_result); i++)
              {
                if (i != 6 && i != 9 && i != 11 && i != 12 && i != 16)
                {
                  if (i == 8)
                  {
                    if (strcmp(sql_row[i], "0") == 0)
                      printf(" not prepaid ");
                    else
                      printf(" prepaid     ");
                  }
                  else
                    printf(" %-13s", sql_row[i]);
                }
              }
              printf("\n");
            }
          }
```

- 우선 billtype을 출력한다.
- query의 결과값의 개수가 0일 경우, NONE을 출력한다.
- customer 별로 출력하여 제공하는데 하나의 customer에 대하여 여러 결제항목이 존재할 수 있다.
- 모든 속성을 출력할 경우, customer의 정보가 여러 번 출력된다.
- 이때는 첫 번째 결과항목이 출력되기 전에만 customer의 정보를 공개하여 가독성을 높인다.
- pre_paid의 값은 0또는 1이므로 0일 경우 후불, 1일 경우 선불임을 보여준다.

```
5
------- TYPE 5 -------
**Generate the bill for each customer for the past month. Consider creating several types of bills.**
Which year? : 2023
Which month? : 5
[RESULT : bills]
<bill type : monthly_pay>
customer_ID : customer_a
name : name_a
account : 1000000001
phone number : 1234
address : private_a
[payment_ID]   [payment_time]      [prepaid?]   [service_ID]  [type]       [status]     [manager]    [sender]     [reciever]   [package_ID]
 payment_a_1   2023-05-31 09:00:00 not prepaid  d_to_a        level_3      delivered    kun          customer_d   customer_a    package_a
 payment_a_2   2023-05-31 09:00:00 not prepaid  e_to_a        level_3      delivered    song         customer_e   customer_a    package_f
 payment_a_3   2023-05-31 09:00:00 not prepaid  c_to_a        level_1      delivered    jung         customer_c   customer_a    package_i

customer_ID : customer_c
name : name_c
account : 1000000003
phone number : 4567
address : private_c
[payment_ID]   [payment_time]      [prepaid?]   [service_ID]  [type]       [status]     [manager]    [sender]     [reciever]   [package_ID]
 payment_c_1   2023-05-31 09:00:00 not prepaid  b_to_c        level_3      delivered    kun          customer_b   customer_c    package_c
 payment_c_2   2023-05-31 09:00:00 not prepaid  a_to_c        level_2      delivered    mee          customer_a   customer_c    package_h

customer_ID : customer_d
name : name_d
account : 1000000004
phone number : 5678
address : private_d
[payment_ID]   [payment_time]      [prepaid?]   [service_ID]  [type]       [status]     [manager]    [sender]     [reciever]   [package_ID]
 payment_d_1   2023-05-31 09:00:00 not prepaid  b_to_d        level_3      delivered    song         customer_b   customer_d    package_d
 payment_d_2   2023-05-31 09:00:00 not prepaid  a_to_d        level_1      delivered    jung         customer_a   customer_d    package_j
```

```
<bill type : credit_card>
customer_ID : customer_b
name : name_b
phone number : 2345
address : private_b
[payment_ID]   [payment_time]      [prepaid?]   [service_ID]  [type]       [status]     [manager]    [sender]     [reciever]   [package_ID]
 payment_b_1   2023-05-16 09:00:00 prepaid      c_to_b        level_3      delivered    kun          customer_c   customer_b    package_b
 payment_b_2   2023-05-16 09:00:00 prepaid      d_to_b        level_2      delivered    mee          customer_d   customer_b    package_g

customer_ID : customer_e
name : name_e
phone number : 6789
address : private_e
[payment_ID]   [payment_time]      [prepaid?]   [service_ID]  [type]       [status]     [manager]    [sender]     [reciever]   [package_ID]
 payment_e_1   2023-05-16 09:00:00 not prepaid  c_to_e        level_3      delivered    song         customer_c   customer_e    package_e


------- SELECT QUERY TYPES -------

       1. TYPE 1
       2. TYPE 2
       3. TYPE 3
       4. TYPE 4
       5. TYPE 5
       0. QUIT
0

C:\Users\User\Desktop\School\5 semseter\Code\DatabaseSystem\Project2\x64\Debug\Project1.exe(프로세스 17728개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

▼ Input 입력 받기

```
printf("Which Truck? : ");
int i = 0;
while ((temp = getchar()) != EOF)
{
  if (temp == '\n' && i != 0)
    break;
  if (temp != '\n')
    truck[i++] = temp;
}
truck[i] = '\0';
```

- input을 입력받는 과정이다. scanf는 공백을 기준으로 입력을 받지만 개행문자를 기준으로 입력을 받기 위해 따로 코드를 구현하였다.
- 개행문자가 처음에 나올 경우, 무시하고 문자열에 입력받지 않는다.
- 이 외에 개행문자가 나올 경우, 입력이 끝난 것이므로 널문자를 넣어서 끝을 나타낸다.

```
printf("When? : ");
            i = 0;
            while ((temp = getchar()) != EOF)
            {
              if (temp == '\n' && i != 0)
                break;
              if (temp != '\n')
                when[i++] = temp;
```

```
            }
            when[i] = '\0';
```

- 트럭의 사고발생시간 또한 정해져야 해당 시간에 있던 package를 고려할 수 있다.
- truck의 운행은 여러 번 있을 수 있기 때문에 특정 사고로 발생으로 인한 데이터를 처리하기 위해서는 사건 발생시간도 필요하다.

▼ query를 통해 얻은 결과값 출력하기

```
sprintf(query, "select distinct customer_ID_sender from carry natural join service where transportation_ID = '%s' and '%s' betw

            state = mysql_query(connection, query);
            if (state == 0)
            {
              printf("[RESULT : (Customer_ID)Sender who had a package on that truck]\n");
              sql_result = mysql_store_result(connection);
              unsigned int num_rows = mysql_num_rows(sql_result);
              if (num_rows == 0)
                printf("[NONE]\n");
              while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
              {
                printf("%s ", sql_row[0]);
                printf("\n");
              }
              mysql_free_result(sql_result);
            }
```

- 입력받은 input들을 query에 넣은 후, query의 결과를 출력한다.
- state가 0일 경우 query가 성공적으로 작동했음을 나타내며, 해당 결과를 출력한다.
- 우선 출력할 내용과 출력할 내용의 속성을 보여준다.
- 만약 출력할 결과값이 하나도 없을 경우, [NONE]을 출력해준다.
- 결과를 저장해놓은 sql_result를 free해준다.

▼ Query

▼ Query 1-1

- printf("**Find all customers who had a package on the truck at the time of the crash.**\n");
- sprintf(query, "
  select distinct customer_ID_sender
  from carry
  natural join service
  where transportation_ID = '%s' and '%s' between carry.from_when and carry.to_when;", truck, when);
  - carry와 service를 natural join해서 해당 사고 트럭이 carry하고 있던 service의 customer_ID_sender를 구한다.

▼ Query 1-2

- printf("**Find all recipients who had a package on that truck at the time of the crash.**\n");
- sprintf(query, "
  select distinct customer_ID_reciever
  from carry
  natural join service
  where transportation_ID = '%s' and '%s' between carry.from_when and carry.to_when;", truck, when);
  - carry와 service를 natural join해서 해당 사고 트럭이 carry하고 있던 service의 customer_ID_reciever를 구한다.

▼ Query 1-3

- printf("**Find the last successful delivery by that truck prior to the crash.**\n");
- sprintf(query, "
  select distinct location_ID_from
  from carry natural join shipment
  where transportation_ID = '%s' and '%s' between carry.from_when and carry.to_when ;", truck, when);

- carry와 shipment를 natural join해서 해당 transportation에 해당하는 출발지(location_ID_from)와 도착지(location_ID_to)를 구한다.
- location_ID_from을 결과값으로 제공한다.

▼ Query 2

- printf("**Find the customer who has shipped the most packages in the past year**\n");
- sprintf(query,
  "SELECT c.customer_ID, c.name,
  COUNT() *AS send_count*
  FROM service s
  JOIN customer c ON s.customer_ID_sender = c.customer_ID
  WHERE s.service_date >= '%04d-01-01' AND s.service_date < '%04d-01-01'
  GROUP BY c.customer_ID, c.name
  HAVING COUNT() = (
  SELECT MAX(send_count)
  FROM(
  SELECT COUNT(*) AS send_count
  FROM service
  WHERE service_date >= '%04d-01-01' AND service_date < '%04d-01-01'
  GROUP BY customer_ID_sender) AS subquery); ",
  cur_year, next_year, cur_year, next_year);
  - SELECT COUNT(*) AS send_count
    FROM service
    WHERE service_date >= '%04d-01-01' AND service_date < '%04d-01-01'
    GROUP BY customer_ID_sender) AS subquery
    - customer_ID_sender별로 입력받은 연도 안에 보낸 service의 개수를 구한다.
  - SELECT MAX(send_count)
    FROM(
    - 위의 query를 통해 그 중 최댓값을 구한다.
  - "SELECT c.customer_ID, c.name,
    COUNT() *AS send_count*
    FROM service s
    JOIN customer c ON s.customer_ID_sender = c.customer_ID
    WHERE s.service_date >= '%04d-01-01' AND s.service_date < '%04d-01-01'
    GROUP BY c.customer_ID, c.name
    - customer_ID별로 입력받은 달 안에 send한 갯수를 구하고 그 중 최댓값과 같은 값을 가지는 것들의 customer_ID와 name을 출력한다.
- 위와 같이 구하는 이유는 최대값을 가지는 customer가 여러 명일 수 있기 때문이다.

▼ Query 3

- printf("**Find the customer who has spent the most money on shipping in the past year.**\n");
- sprintf(query,
  "SELECT c.customer_ID, c.name
  FROM service s
  JOIN payment p ON s.payment_ID = p.payment_ID
  JOIN customer c ON p.customer_ID = c.customer_ID
  JOIN service_type_cost stc ON s.service_type = stc.service_type
  WHERE s.service_date >= '%04d-01-01' AND s.service_date < '%04d-01-01'
  GROUP BY c.customer_ID, c.name
  HAVING SUM(stc.service_cost) = (
  SELECT MAX(total_payment)
  FROM(
  SELECT SUM(stc.service_cost) AS total_payment
  FROM service s

JOIN payment p ON s.payment_ID = p.payment_ID
JOIN service_type_cost stc ON s.service_type = stc.service_type
WHERE s.service_date >= '%04d-01-01' AND s.service_date < '%04d-01-01'
GROUP BY p.customer_ID) AS subquery);",
cur_year, next_year, cur_year, next_year);

- SELECT SUM(stc.service_cost) AS total_payment
  FROM service s
  JOIN payment p ON s.payment_ID = p.payment_ID
  JOIN service_type_cost stc ON s.service_type = stc.service_type
  WHERE s.service_date >= '%04d-01-01' AND s.service_date < '%04d-01-01'
  GROUP BY p.customer_ID) AS subquery);",
  cur_year, next_year, cur_year, next_year);
  - 위의 query는 service를 payment와 service_type_cost와 join해서 customer별로 입력받은 달 안에 지불해야하는 cost값을 합한다.
- SELECT MAX(total_payment)
  FROM(
  - 위의 query를 통해 구한 값들 중 가장 높은 cost의 합을 구한다.
- SELECT c.customer_ID, c.name
  FROM service s
  JOIN payment p ON s.payment_ID = p.payment_ID
  JOIN customer c ON p.customer_ID = c.customer_ID
  JOIN service_type_cost stc ON s.service_type = stc.service_type
  WHERE s.service_date >= '%04d-01-01' AND s.service_date < '%04d-01-01'
  GROUP BY c.customer_ID, c.name
  HAVING SUM(stc.service_cost) = (
  - 위의 query를 통해 service와 payment와 customer를 join해서 customer별로 입력받은 달에 지불해야하는 cost의 합을 구하고, 이전에 구했던 최댓값을 가지는 customer의 ID와 이름을 출력한다.

- 위와 같이 구하는 이유는 최대값을 가지는 customer가 여러 명일 수 있기 때문이다.

▼ Query 4

- printf("**Find those packages that were not delivered within the promised time.**\n");
- sprintf(query, "
  SELECT p.package_ID
  FROM package p JOIN service s ON p.package_ID = s.package_ID
  JOIN(
  SELECT package_ID, MAX(to_when) AS latest_locates_time
  FROM locates GROUP BY package_ID) l
  ON p.package_ID = l.package_ID
  WHERE l.latest_locates_time > s.promised_time;");
  - SELECT package_ID, MAX(to_when) AS latest_locates_time
    FROM locates GROUP BY package_ID) l
    - 위에서 l은 마지막 도착시간이다. locates에서 가장 마지막 장소는 시작시간과 종료시간이 같으므로 이 점을 활용하여 상품 배송 도착시간을 측정한다. 그리고 해당시간을 latest_locates_time이라고 한다.
  - package p JOIN service s ON p.package_ID = s.package_ID
    JOIN(
    - package와 위의 결과값을 join하여 package들의 latest_locates_time을 구한다.
  - SELECT p.package_ID
    …..
    WHERE l.latest_locates_time > s.promised_time;");
    - 그 중 latest_locates_time(도착시간)이 promised_time(약속 시간)보다 늦은 경우를 결과로 출력한다.

▼ Query 5

- printf("**Generate the bill for each customer for the past month. Consider creating several types of bills.**\n");
- sprintf(query, "
  SELECT c.*, p.*, s.*
  FROM payment p
  NATURAL JOIN service s
  NATURAL JOIN customer c
  WHERE p.payment_type = '%s'
  AND s.service_date >= '%04d-%02d-01' AND s.service_date < '%04d-%02d-01';", bill_type, year, cur_month, year, next_month);
  - FROM payment p
    NATURAL JOIN service s
    NATURAL JOIN customer c
    - payment와 service와 customer를 join하여 payment가 같은 경우끼리, customer별로 bill을 제공한다. 입력받은 달에 있는 service만 선택한다.
    - 모든 속성들을 다 출력하게한 후, 특정 속성들만 선택하여 형식을 맞추었다.