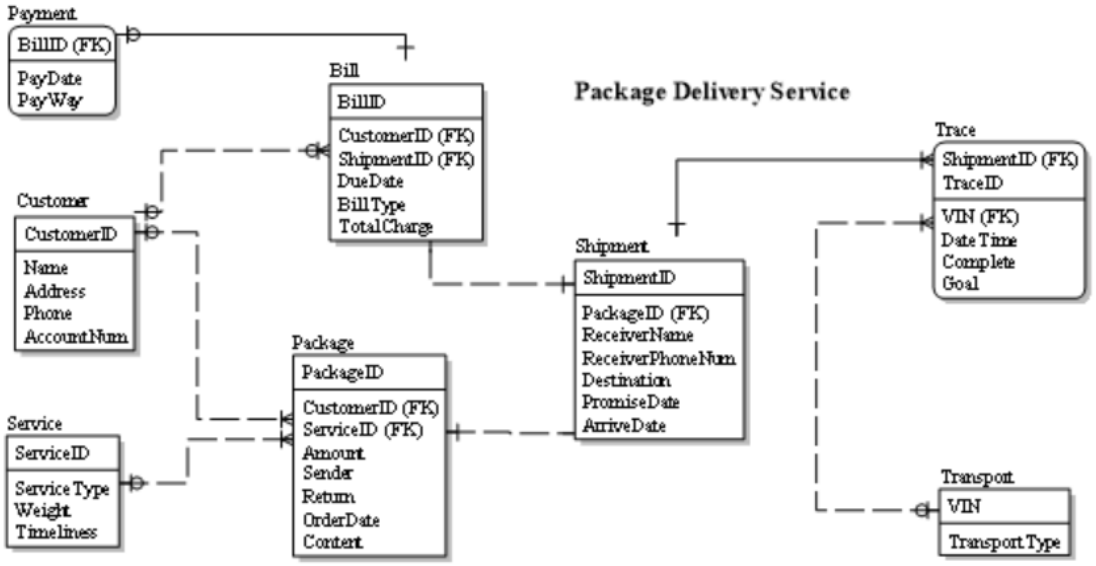
DB 프로젝트2 <20191612 윤기웅>



1) BCNF decomposition

효율적인 DB 설계를 위해서 bcnf를 만족하는 지 확인하고 만약 만족하지 않는 경우 decomposition을 통해서 bcnf 형태로 맞춰준다. BCNF분해를 위해서 각각의 FD(a->b)에 대해서 non-trivial하면서도 슈퍼키가 아닌 a가 존재하는 지 확인한다. 각각의 entity에 대해서 한 번 살펴보도록 한다.

Customer : Customer를 살펴보면 프라이머리 키에 해당하는 CustomerID를 제외한 속성에는 다른 fd가 존재하지 않는다. 회원들 간에 동명이인이 있을 수 있고 주소가 같거나 같은 집에 사는 회원들이 있을 수 있기 때문에 Name이나 Address는 추가적인 fd를 형성하지는 않는다.

Service : Service entity를 보면 ServiceID를 제외한 속성에 대한 fd가 존재하지 않는다. 서비스 타입, 택배의 무게, Timeliness을 종합적으로 고려해서 ServiceID가 부여되기 때문이다. 특정 무게를 갖는 택배가 특정 Timeliness를 갖는다고 할 수 없고 서비스 타입을 결정하지도 않는다. 이는 다른 attribute인 서비스 타입이나 Timeliness에 대해서도 마찬가지다. 따라서 BCNF 조건을 모두 충족하고 있다.

Bill : 이 entity를 보면 BillID를 제외한 다른 fd가 존재하지 않는다. BillID는 프라이머리

키라서 BCNF의 만족 조건을 해치지 않는다. 따라서 BCNF를 만족한다.

Shipment : 프라이머리 키인 ShipmentID를 제외하고 PackageID로 인해서 발생하는 fd가 존재한다. PackageID, Destination->DeliverFee의 fd가 또한 존재하는데 이 때 BCNF의 조건을 충족하지 못한다. 그래서 DeliverFee는 Bill의 TotalCharge와 일치하기 때문에 제거해주기로 한다. 구태여 같은 값을 두 개의 entity에 저장할 필요가 없다. 이전의 설계에는 없었던 ArriveDate를 추가해서 언제 실제 패키지가 도착했는 지 기록할 수 있게 한다.

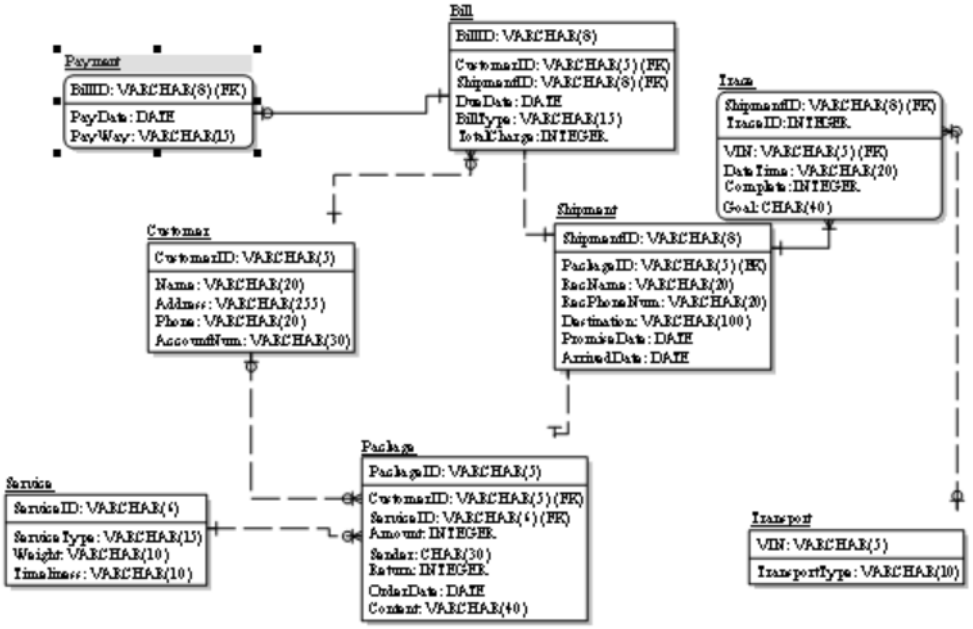
Package : 프라이머리 키인 PackageID를 제외하고 다른 속성에서 비롯하는 fd가 없다. cutomerID 혹은 serviceID로부터 비롯되는 fd가 있다고 생각할 수도 있지만 특정 제품을 여러 번 주문할 수도 있기 때문에 fd가 성립하지 않는다. 이전에 없었던 content 속성을 추가해서 어떠한 물건인지 확인하고 위험물질은 아닌 지 분간이 되도록 한다.

Trace : 프라이머리 키인 ShipmentID, TraceID에서 비롯한 fd를 제외하고는 다른 fd가 존재하지 않아서 BCNF가 성립하게 된다. 또한, 기존의 PlaceDateTime에서 장소까지 저장했지만 DateTime으로 바꾸면서 장소는 기록하지 않기로 했다. 어차피 Goal에 장소가 저장되기 때문이다.

Transport : 고유의 운송수단들에 부여되는 VIN이 프라이머리 키로 존재하고 이외에는 fd가 없음으로 이 entity는 BCNF를 만족한다.

Payment : 이전에 없었던 entity이고 고객의 지불내역에 대한 정보를 저장하기 위한 entity이다. bill 하나에 대해서 하나씩 생성되고 PayDate에 고객이 지불한 날짜가 기록되고 PayWay에 지불 방법이 기록된다. 프라이머리 키로 설정된 BillID를 제외하고는 다른 속성으로부터 비롯되는 fd가 없기 때문에 BCNF 조건을 만족한다. 한 고객이 여러가지 지불 방법을 택할 수 있어서 이에 대한 fd가 존재하지 않는다.

2) Physical schema (ERWin)



- **Customer:**

  CustomerID: 'C0072'처럼 고객 아이디가 저장되고 varchar(5)로 정한다. customer entity의 프라이머리 키 이라서 not null 의 특징을 갖는다.

  Name: 'Kiwoong Yoon'처럼 고객의 영어 이름이 저장되기 때문에 varchar(20)으로 정해서 긴 이름 또한 저장 되도록 한다. 이름이 없는 이가 없어서 not null으로 설정한다.

  Address : "123, Hangang-daero, Yongsan-gu, Seoul"와 같이 고객의 주소가 들어가도록 하기 위해 varchar(255)로 타입을 정한다. 또한, 고객 회원가입을 할 때 꼭 주소를 기입하지 않을 수 있으니 null 값을 허용한다.

  Phone : 고객의 전화번호가 '010-1234-1234'와 같이 저장되기 때문에 varchar(20)으로 저장한다. 그러면 집 전화와 인터넷 전화, 해외 전화번호 까지도 저장할 수 있다. 본인 확인 절차 등에 사용될 수 있으니 not null으로 설정한다.

  AccountNum : '302-0149-894141'와 같은 형식의 계좌번호를 저장한다. 각각의 은행의 계좌번호에 따라서 자리수가 다를 수 있으니 varchar(30)로 설정한다. 계좌를 통해서 결제를 하지 않는 사람도 있으니 필수로 기입할 필요는 없게 한다.

- **Shipment**

  shipmentID : S2200144의 형태로 저장이 되고 varchar(8)으로 저장하고 프라이머리 키이기 때문에 null 값은 저장될 수 없다.

PackageID: Package에서 온 외래 키이기 때문에 같은 도메인을 갖도록 설정한다.

RecName: 받는 사람의 이름이기 때문에 varchar(20)으로 정하고 null 값을 갖지 못하도록 정한다.

RecPhoneNum: 받는 이의 전화번호가 저장되기 때문에 varchar(20)으로 설정해서 인터넷 전화나 집 전화 또한 포함되도록 설정한다. null 값을 갖지 못하도록 정한다.

Destination: 수령인이 물건을 받을 장소이다. 다소 긴 주소가 들어올 수 있기 때문에 varchar(100)으로 정하고 null 값을 갖지 못하도록 정한다.

PromiseDate: 물건이 언제 도착할 지 고객에게 알려주는 날짜로 DATE 타입을 지정한다.

ArrivedDate: 실제로 도착한 날을 저장하는 것으로 DATE 타입으로 지정한다. 조회 시점에서 아직 패키지가 수령인에게 도착하지 않을 수 있으니 null 값을 허용한다.

- **Package**

PackageID : P0110처럼 아이디가 저장되도록 varchar(5)로 지정하고 프라이머리 키 이라서 null 값은 갖지 않는다.

CustomerID: 고객의 아이디를 저장하는 외래 키이고 varchar(5)으로 지정한다.

ServiceID: 서비스의 종류의 아이디를 저장하는 외래 키이고 "Ser-03"과 같은 형태이니 varchar(6)의 형태로 저장한다.

Amount: 물건이 몇 개가 패키지에 있는 지 저장하기 때문에 integer 형태를 갖는다.

Sender: 보내는 이의 이름이나 단체 등이 저장되고 주로 회사 명이 "Samsung"처럼 저장되도록 varchar(30)으로 설정해서 긴 이름이 들어올 수 있게 지정한다.

Return: 반송되는 물품인지 여부를 확인하는 것으로 integer 타입을 갖고 만약 1이면 반송되는 물품, 0인 경우는 반송 물품이 아닌 것으로 정한다.

OrderDate: 고객이 주문한 날짜로 "2022-12-22"와 같은 형태로 저장되기 때문에 DATE 타입으로 지정한다.

Content: 어떤 물건이 들어있는 지 확인이 되도록 하고 취급 시 주의해야 하는 지 보내는 이가 적을 수 있게 해야 하니 varchar(40)으로 지정한다. null 값을 갖지 못하도록 설정해서 위험한 것인지 구분되도록 한다.

- **Bill**

BillID: 여러 bill 들에 대해서 구분이 되도록 B0000011처럼 앞의 B와 7자리 수가 합

쳐지도록 한다. 그래서 varchar(8) 타입을 사용한다. 그리고 null 값을 갖지 못하도록 한다.

CustomerID: Customer의 프라이머리 키를 외래 키로 갖고 와서 사용하기 때문에 같은 타입인 varchar(5)로 지정한다.

ShipmentID: Shipment의 키를 외래 키로 갖고 와서 사용하기 때문에 이전의 타입인 varchar(8) 타입을 지정한다.

DueDate: 언제까지 배송비를 지불해야 하는 지 저장하는 속성으로 Date 타입으로 지정한다. 이는 고객이 실제로 배송비를 지불한 날짜와는 다르다.

BillType: 청구 방식을 나타내는 속성이라서 달 마다 배송비를 지불하는 고객의 경우 month, 그냥 그때그때 카드로 결제하는 경우는 card, 이미 지불한 경우는 already paid로 저장한다. 그래서 varchar(15)로 지정한다.

TotalCharge: 고객이 지불해야 하는 돈이고 배송비는 0원에서부터 보통 3500원이고 산간 지역이나 섬의 경우 추가적으로 배송비가 붙으니 INTEGER 타입을 사용한다.

- **Service**

ServiceID: 패키지의 타입이나 패키지의 무게 그리고 Timeliness으로 정해지는 서비스 아이디이다. Ser-02의 형태로 저장되기 때문에 varchar(6)으로 설정한다.

ServiceType: "flat", "little box" , "huge box" 등으로 패키지의 크기에 맞는 타입을 저장해야 하니까 varchar(15)으로 설정한다. 서비스 타입이 없는 패키지는 없으니 null 값을 허용 되지 않도록 만든다.

Weight: 패키지의 무게 범위를 정해서 "light", "middle", "heavy" 중 하나로 저장하기 위해서 varchar(10)으로 설정해서 null 값을 허용하지 않게 만든다. 무게의 기준은 회사마다 다를 수 있지만 여기에서는 3키로 미만은 light , 3키로 이상 8키로 미만은 middle, 그 이상은 heavy로 정한다. 이 범위에 벗어나는 패키지는 없기 때문에 null 값이 허용되지 않도록 한다.

Timeliness: "Fast", "Normal" 중 하나로 정해지기 때문에 varchar(10)으로 지정하고 null 값은 허용되지 않도록 설정한다.

- **Trace**

ShipmentID: Shipment의 아이디를 외래 키로 사용하고 Shipment 정보가 사라지면 이 또한 의미가 사라지니까 on delete cascade를 적용한다. 타입은 varchar(8)으로 그대로 쓴다.

TraceID: 하나의 Shipment가 여러 과정을 거쳐서 운송되기 때문에 trace 아이디를 통해서 각 과정을 관리한다. INTEGER 형태로 지정되니까 1부터 순서에 맞게 숫자가 부여된다.

VIN: Transport 에서 사용되는 VIN을 갖고 와서 외래 키로 사용한다. 같은 형식을 쓸 수 있도록 int로 설정한다.

DateTime: 현재 Trace에서 날짜 및 시간 그리고 위치를 나타내는 속성이다.

"2022-12-12 10:00:00"의 형식으로 저장하기 위해서 VARCHAR(20)을 지정한다.

Complete: 하나의 Shipment에 해당하는 특정 Trace가 어느 상태에 있는 지 숫자를 이용해서 표현한다. 0은 아직 차례가 오지 않은 경우, 1은 현재 수행중인 경우, 2는 성공적으로 수행된 경우, 그리고 3은 운송 도중 사고나 예상하지 못한 사건이 발생한 경우를 의미한다. INTEGER의 타입을 갖도록 지정한다.

Goal: 현재 Trace의 목적지를 나타내고 최종적으로 받는 이의 주소가 기록되거나 주된 물류 창고나 공항 등을 기록한다. "Incheon Airport", "Seoul ware house" 등이 해당하니 varchar(40)으로 설정한다. null 값은 허용되지 않도록 한다.

- **Transport**

  VIN: 원래는 자동차의 고유 번호로 쓰이는 VIN을 다른 운송수단의 구분도 하기 위해서 이 데이터에서 사용한다. 0230 혹은 0120과 같은 형식을 갖으니 int로 지정한다. 또한, null 값이 들어오지 못 하게 설정한다.

  TransportType: "Truck", "Ship", "Airplane"과 같은 운송수단의 이름이 저장되어야 하니 varchar(10)으로 지정한다. 또한, null 값이 들어오지 못하게 설정한다.

- **Payment**

  BillID: 어떤 bill에 대해서 지불한 내역인 지 확인이 가능해야 하니까 BillID를 외래 키로 사용하고 같은 타입을 사용하기에 varchar(8)을 사용한다. 이는 외래 키이면서 프라이머리 키로 사용되니까 만약 bill이 삭제되면 이 또한 제거될 수 있게 on delete cascade를 적용한다.

  PayDate: Bill에 대한 고객의 지불이 이루어진 날을 저장해야 하니 DATE 타입을 적용한다. null 값을 허용하지 않는다.

PayWay: Bill에 대한 고객의 지불 방식을 나타낸다. "cardpay", "cashpay", "onlinepay"등이 해당하고 null 값을 허용하지 않는다. 타입은 VARCHAR(15)으로 설정한다.

3) ODBC and Queries

* 아래와 같은 쿼리문을 만들어서 table을 만들고 데이터들을 넣었다.

```
create table package(
    PackageID varchar(5),
    CustomerID varchar(5),
    ServiceID varchar(6),
    Amount int not null ,
    Sender varchar(30) not null ,
    IsReturn int,
    OrderDate DATE,
    Content varchar(40),
    primary key (PackageID));
alter table package add constraint fk_pac_cust foreign key(CustomerID) references customer(CustomerID) on delete set null ;
alter table package add constraint fk_pac_ser foreign key(ServiceID) references service(ServiceID) on delete set null ;


create table shipment (
    ShipmentID varchar(8),
    PackageID varchar(5),
    RecName  varchar(20) not null ,
    RecPhoneNum varchar(20) not null ,
    Destination varchar(100) not null ,
    PromiseDate DATE,
    ArrivedDate DATE,
    primary key(ShipmentID));
alter table shipment add constraint fk_ship_pac foreign key(PackageID) references package(PackageID)on delete set null;


create table bill(
    billID varchar(8),
    CustomerID varchar(5),
    ShipmentID varchar(8),
    DueDate DATE ,
    BillType varchar(15) not null ,
    TotalCharge int ,
```

```
INSERT INTO customer VALUES ("C0001","Bong Joon Ho","123, Hangang-daero, Yongsan-gu, Seoul","010-9868-1585","115-373-036544");
INSERT INTO customer VALUES ("C0002","Park Shin Hye","456, Daeheung-ro, Jongno-gu, Seoul","010-4909-1891","136-640-341895");
INSERT INTO customer VALUES ("C0003","Lee Min Ho","789, Dongmun-gil, Jung-gu, Incheon","010-1688-8194","410-714-259072");
INSERT INTO customer VALUES ("C0004","Park Bo Gum","321, Seomjingang-ro, Nam-gu, Gwangju","010-2077-3211","333-611-074209");
INSERT INTO customer VALUES ("C0005","Jun Ji Hyun","654, Haeundae Beach Road, Haeundae-gu, Busan","010-8241-9681","613-829-351868");
INSERT INTO customer VALUES ("C0006","Park Ji Sung","987, Sejong-daero, Jung-gu, Sejong City","010-7932-7284","119-552-987205");
INSERT INTO customer VALUES ("C0007","Kim Soo Hyun","70, Yeongmun-ro, Pogok-eup, Cheoin-gu, Yongin-si, Gyeonggi-do","010-7895-1341","650-859-168957");
INSERT INTO customer VALUES ("C0008","Bae Suzy","135, Daewangpangyo-ro, Bundang-gu, Seongnam-si","010-8423-9175","393-091-549401");
INSERT INTO customer VALUES ("C0009","Song Joong Ki","579, Dalseo-daero, Dalseo-gu, Daegu","010-6402-9022","228-382-000829");
INSERT INTO customer VALUES ("C0010","Lee Jong Suk","72-67, Saeteo-gil, Sang-myeon, Gapyeong-gun, Gyeonggi-do","010-7515-0082","110-147-461538");
INSERT INTO customer VALUES ("C0011","Hasung Kim","194-30, Boreumgol-gil, Jojong-myeon, Gapyeong-gun, Gyeonggi-do","010-9054-6569","105-447-469351");
INSERT INTO customer VALUES ("C0012","Ha Ji Won","813, Songpa-daero, Songpa-gu, Gangnam-gu, Seoul","010-0203-2884","107-710-578096");
INSERT INTO customer VALUES ("C0013","Park Seo Joon","234, Gupyeongsinheung-gil, Buk-myeon, Jeongeup-si, Jeollabuk-do","010-4952-1561","810-293-023759");
INSERT INTO customer VALUES ("C0014","Park Chan Wook ","10-18, Seocho-daero, Seocho-gu, Seoul","010-4681-8725" , "110-231-324224");

insert into service values ("Ser-01","flat","light","Normal")    ;
insert into service values ("Ser-02","flat","light","Fast");
insert into service values ("Ser-03","flat","middle","Normal");
insert into service values ("Ser-04","flat","middle","Fast");
insert into service values ("Ser-05","flat","heavy","Normal");
insert into service values ("Ser-06","flat","heavy","Fast");
insert into service values ("Ser-07","little box","light","Normal");
insert into service values ("Ser-08","little box","light","Fast");
insert into service values ("Ser-09","little box","middle","Normal");
insert into service values ("Ser-10","little box","middle","Fast");
insert into service values ("Ser-11","little box","heavy","Normal");
insert into service values ("Ser-12","little box","heavy","Fast");
insert into service values ("Ser-13","huge box","light","Normal");
insert into service values ("Ser-14","huge box","light","Fast");
insert into service values ("Ser-15","huge box","middle","Normal");
insert into service values ("Ser-16","huge box","middle","Fast");
insert into service values ("Ser-17","huge box","heavy","Normal");
insert into service values ("Ser-18","huge box","heavy","Fast");
```

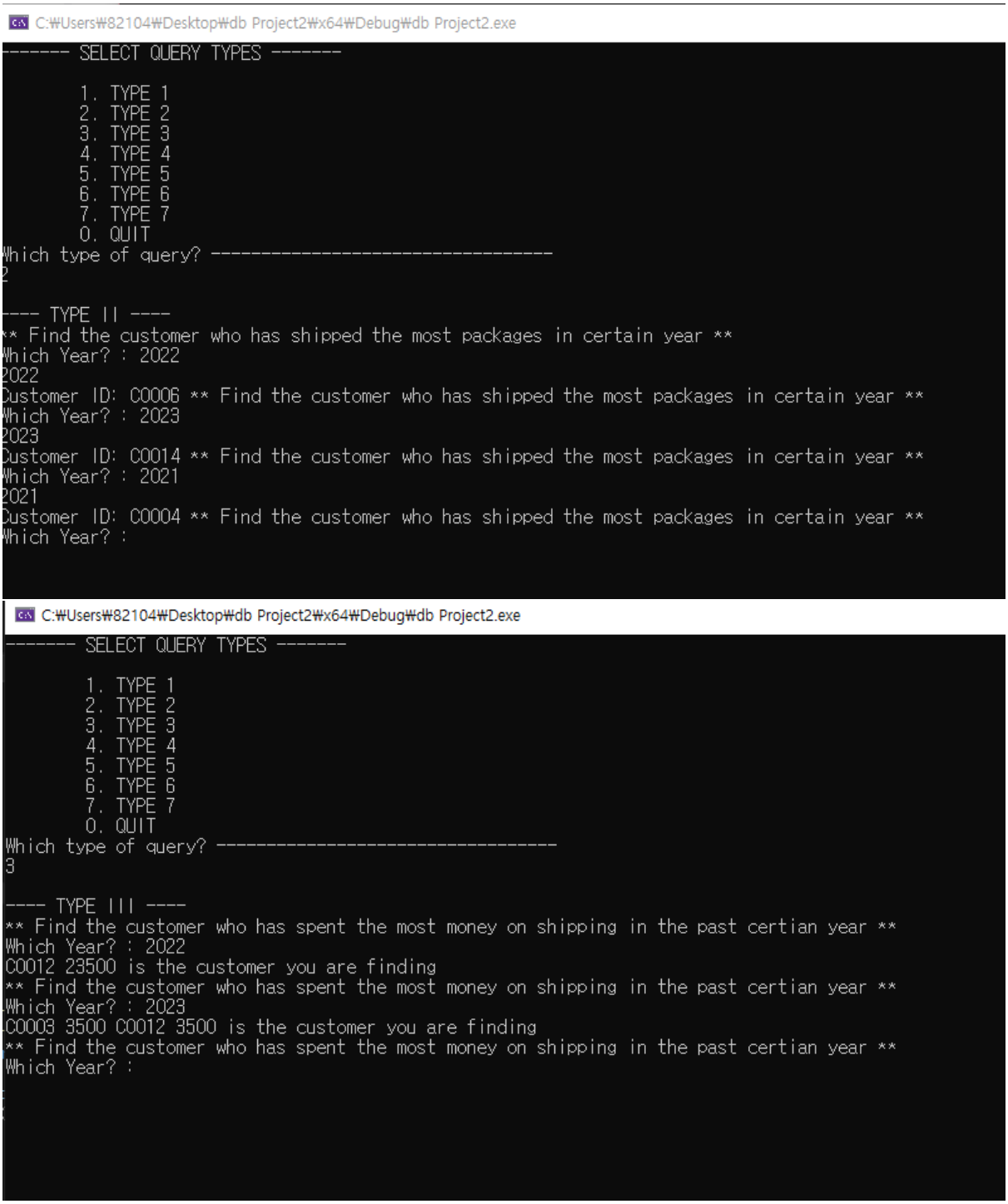
*아래의 사진은 쿼리문을 실행한 경우이다. customer를 구해야 하는 경우 같은 이름의 고객이 있을 수 있어서 고객의 ID를 출력하는 것으로 구현했다.

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

        1. TYPE 1
        2. TYPE 2
        3. TYPE 3
        4. TYPE 4
        5. TYPE 5
        6. TYPE 6
        7. TYPE 7
        0. QUIT
Which type of query? ---------------------------------
1

---- TYPE 1 ----
Input the number of Transport : 1721
1721
---- Subtypes in TYPE 1----
        1. TYPE 1-1
        2. TYPE 1-2
        3. TYPE 1-3
Which type of query? 1

---- TYPE 1-1 ----
** Find all customers who had a package on the truck at the time of the crash. **
Customer ID: C0006 | C0002 |


---- Subtypes in TYPE 1----
        1. TYPE 1-1
        2. TYPE 1-2
        3. TYPE 1-3
Which type of query? 2

---- TYPE 1-2 ----
** Find all recipients who had a package on the truck at the time of the crash. **
 recipients name :  Junbin Kwak | Byunggun Lee |
```

```
---- Subtypes in TYPE 1----
        1. TYPE 1-1
        2. TYPE 1-2
        3. TYPE 1-3
Which type of query? 3

---- TYPE 1-3 ----
** Find the last successful delivery by that truck prior to the crash. **
Last Successful Delivery on: receiver | 2021-05-05


---- Subtypes in TYPE 1----
        1. TYPE 1-1
        2. TYPE 1-2
        3. TYPE 1-3
Which type of query? 0

C:₩Users₩82104₩Desktop₩db Project2₩x64₩Debug₩db Project2.exe(프로세스 14976개)이(가) 종료되었습니다(코드: 0개
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔
이 창을 닫으려면 아무 키나 누르세요...
```

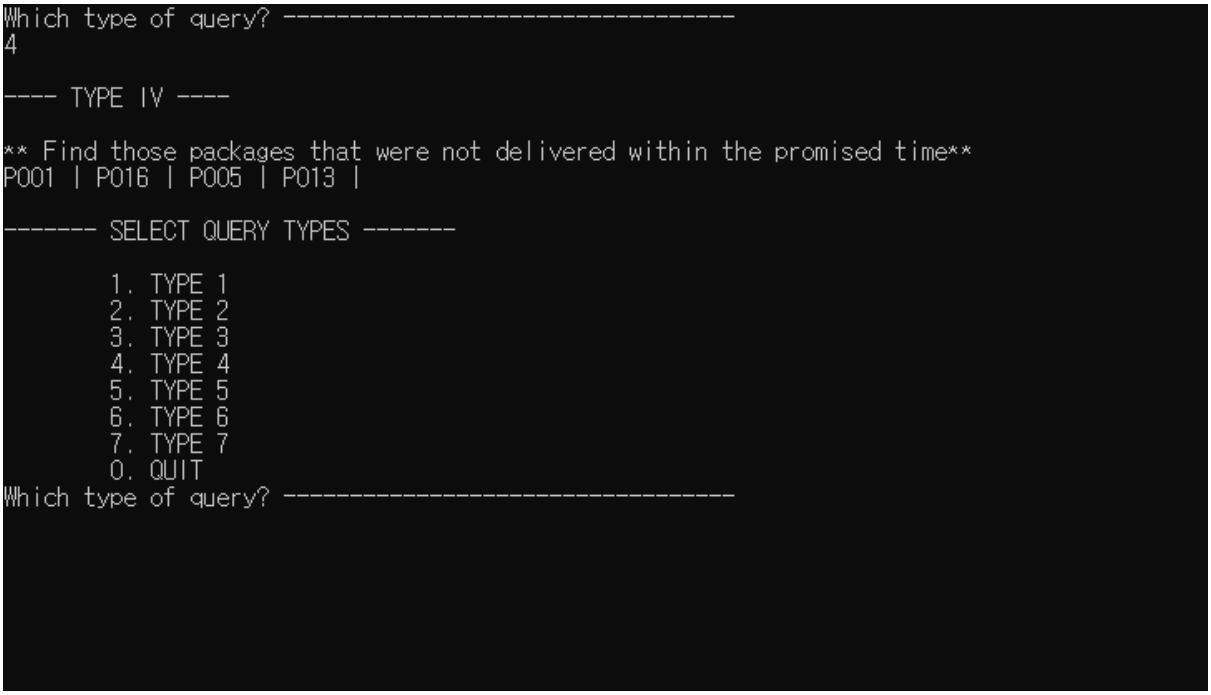
*특정 연도에 가장 많은 패키지 서비스를 이용한 고객의 아이디와 특정 년도에 제일 많

은 비용을 쓴 고객의 아이디를 출력하도록 구현했다.



*

* 패키지 중에서 예상 도착일 보다 더 늦게 도착한 패키지들의 아이디를 출력하도록 했다.



* 코드는 아래와 같이 미리 작성된 sql문을 실행할 수 있도록 했다. type 변수를 어떤 수로 정할 지 사용자로부터 입력을 받고, 그 수에 맞는 코드를 실행시킨다. 1~5번사이의 수를 입력받으면 작성된 sql문이 실행되고 0이 입력받으면 프로그램이 종료된다.

1번을 입력받으면 추가적으로 1~3사이의 수를 입력받아서 다시 파손된 트럭에 관련된 데이터를 얻을 수 있다. 2,3번을 입력받으면 추가적으로 년도를 입력받고 그 년도에 대한 최다 사용자, 최대 금액 지불자를 출력한다. 4번은 정해진 도착일정보다 늦어진 패키지에 대해서 데이터를 얻을 수 있게 한다.

```
else if (type == 3)
{
    printf("---- TYPE III ----\n");
    int year2;
    while (1)
    {
        printf("++ Find the customer who has spent the most money on shipping in the past certian year ++\n");
        printf("Which Year? : ");
        scanf("%d", &year2);
        char yearstr[5];
        sprintf(yearstr, "%d", year2);
        char query3[500] = "SELECT t1.customerID, SUM(t1.TotalCharge) AS chargesum FROM bill t1 JOIN payment t2 ON t1.billID = t2.billID WHERE YEAR(t2.PayDate) = ";
        strcat(query3, yearstr);
        strcat(query3, " GROUP BY t1.customerID HAVING SUM(t1.TotalCharge)=(SELECT MAX(total_chargesum) FROM ( SELECT SUM(t1.TotalCharge) AS total_chargesum  FROM bill t1 JOIN payment t2 ON t1.billID = t2.billID WHERE YEAR
        strcat(query3, yearstr);
        strcat(query3, " GROUP BY t1.customerID) AS subquery) ORDER BY chargesum DESC");
        int check3 = 0;
        check3 = mysql_query(connection, query3);
        if (check3 == 0)
        {
            sql_result = mysql_store_result(connection);
            while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
            {
                printf("%s %s ", sql_row[0], sql_row[1]);
            }

            mysql_free_result(sql_result);
        }
        printf("is the customer you are finding \n");

    }
}//type = 3
```