cse4110 – Database System

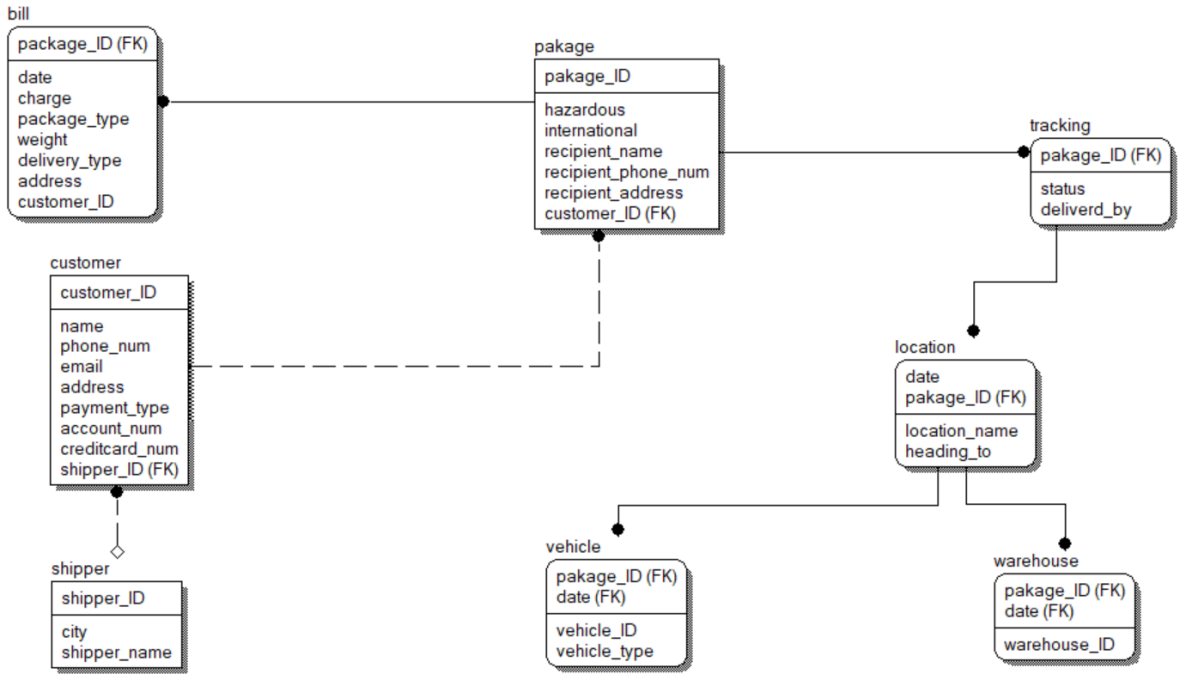
project 2

컴퓨터공학과 20161640

정동혁

1. BCNF form

1-1. Before BCNF logical schema



a). bill

bill entity에서의 functional dependency는 {package_ID -> date, charge, package_type, weight, delivery_type, address, customer_ID}만이 있다. 여기서 package_ID는 bill entity의 superkey이므로 bill entity는 BCNF form인 것을 알 수 있다.

b). customer

customer entity에서의 functional dependency는 {cutomer_ID -> name, phone_num, email, address, payment_type, account_num, creditcard_num, shipper_ID}만이 있다. 여기서 customer_ID는 customer entity의 superkey이므로 customer entity는 BCNF form인 것을 알 수 있다.

c). shipper

shipper entity에서의 functional dependency는 {shipper_ID -> city, shipper_name}만이 있다. 여기서 shipper_ID는 superkey이므로 shipper entity는 BCNF form인 것을 알 수 있다.

d). package

package entity에서의 fucntional dependency는 {package_ID -> hazardous, international, recipient_phone_num, recipient_address, customer_ID}만이 있다. 여기서 package_ID는 package entity의 superkey이므로 customer entity는 BCNF form인 것을 알 수 있다.

e). tracking

tracking entity에서의 functional dependency는 {packge_ID -> status, deliverd_by}만이 있다. 여기서 package_ID는 tracking entity의 superkey이므로 tracking entity는 BCNF form인 것을 알 수 있다.

f). location

location entity에서의 functional dependency는 {package_ID,date -> location,heading_to}만이 있다. 여기서 package_ID, data는 location entity의 superkey이므로 location entity는 BCNF form인 것을 알 수 있다.

g). vehicle

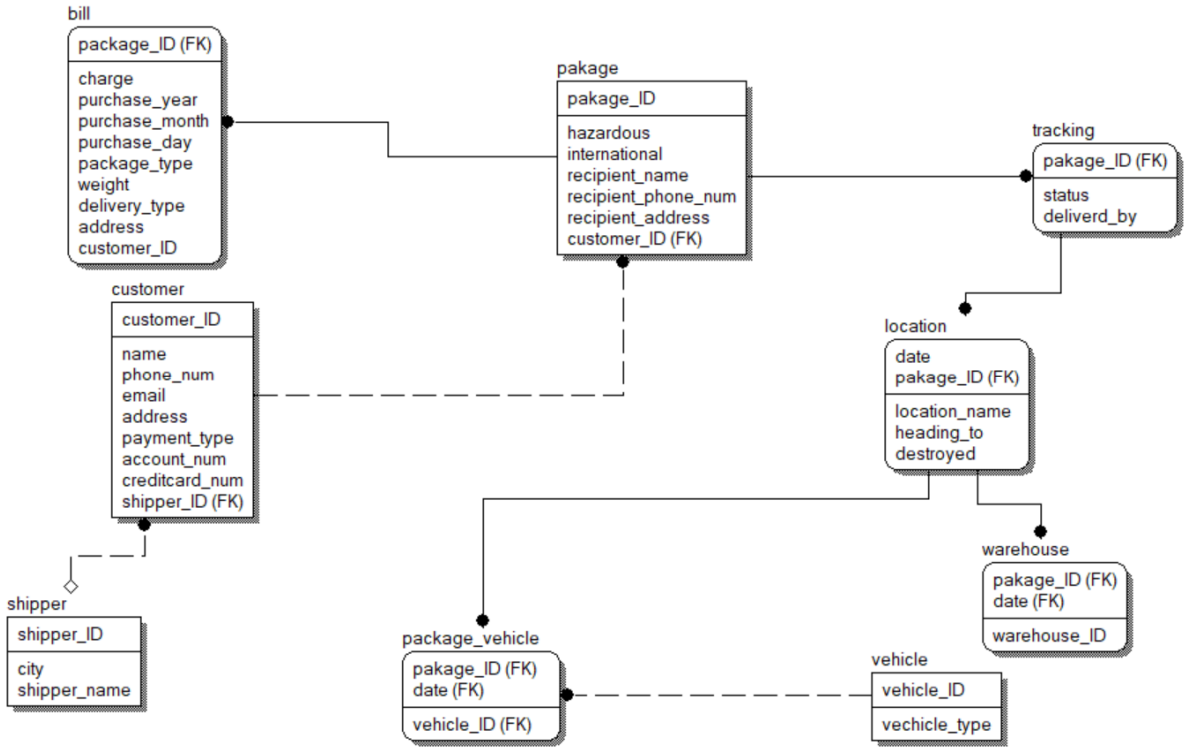
vehicle entity에서의 functional dependency는 {pakage_ID,date ->vehicle_ID,vehicle_type}, {vehicle_ID -> vehicle_type}등이 있다. 여기서 첫 번째 FD는 package_ID, date가 superkey이므로 BCNF form을 만족하지만 두 번째 FD에서는 vehicle_ID가 superkey가 아니고 vehicle_ID -> vehicle_type이 trivial한 FD가 아니므로 BCNF form이 아니다. 그러므로 decompose가 필요하다.

h). warehouse

warehouse entity에서의 functional dependency는 {package_ID,date -> warehouse_ID}만이 있다. 여기

서 {package_ID, date}는 warehouse entity의 superkey이므로 warehouse entity는 BCNF form인 것을 알 수 있다.

1-2. After BCNF logical schema



(BCNF중간과정에 query구현을 위한 추가적인attribute(bill.(purchase_year, purchase_month, purchase_day), location.destroyed)를 추가하였다.

1-2-1 BNCF process

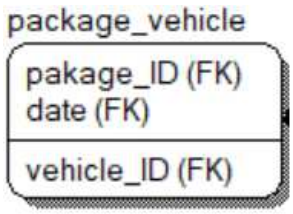
위에서 vehicle entity가 BCNF form을 만족하지 않음을 확인하였다. 그러므로 vehicle entity를 package_vehicle, vehicle entity로 BCNF decomposition을 진행하였다. 우선 언급하였던 vehicle entity의 funtional dependency를 다시보면 다음과 같다.

a). package_ID, date -> vehicle_ID, vehicle_type

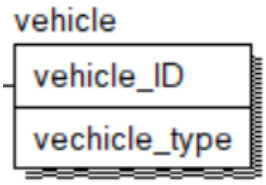
b). vehicle_ID -> vehicle_type

FD는 다음과 같으므로 b번째 dependency가 BCNF form을 만족하기 위하여 다음과 같이 decompose를 해주었다.

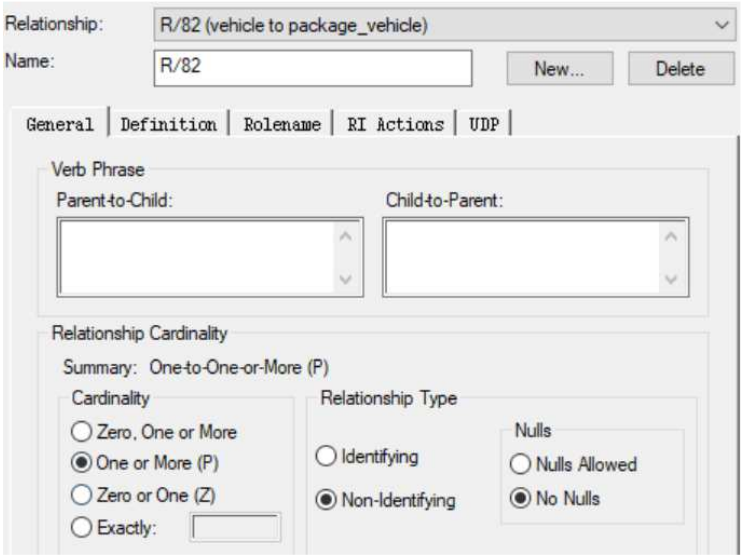
(1). package_vehocle



(2). Vehicle



(3). relation



위와 같이 one-to-many relation으로 두개의 entity를 decompose를 해줌으로서 package_vehicle entity에 존재하는 FD는 다음과 같다.

package_ID,date -> vehicle_ID

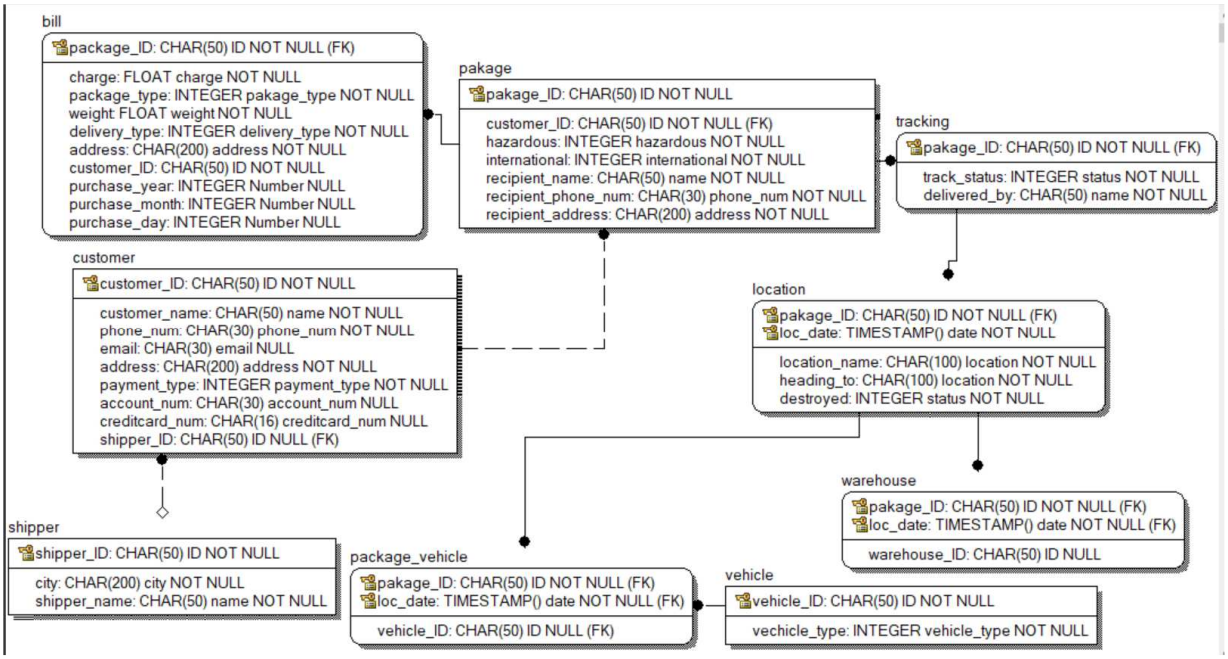
여기서 package_ID, date는 package_vehicle entity의 super key이므로 BCNF form을 만족한다.

또한 위와 같이 decompose를 해줌으로서 vehicle entity에 존재하는 FD는 다음과 같다.

vehicle_ID -> vehicle_type

여기서 vehicle_ID는 vehicle entity의 super key이므로 BCNF form을 만족한다.

2. Physical model



physical model에 모든 attribute에 적합한 domain을 제작하였으므로 Domain dictionary를 통해 domain설정을 설명하겠다.

5-1). account_num

일반적으로 계좌번호는 20자리를 넘지 않고 은행명을 언급하기 위해 CHAR(30)으로 선언하였다. 계좌번호 이외의 결제방식을 이용할 수 있으므로 NULL값을 허용해주었다.

5-2). address

주소의 경우 CHAR(200)으로 최대길이를 200으로 지정해주었다. 주소가 존재하지 않을 경우 배송이 불가능하므로 NOT NULL로 해주었다.

5-3). charge

배송비를 의미하는 charge는 해외의 경우 2.99$와 같이 소수점자리를 가질 수 있으므로 FLOAT형으로 지정해주었다. 금액은 없을수 없으므로 NOT NULL로 해주었다.

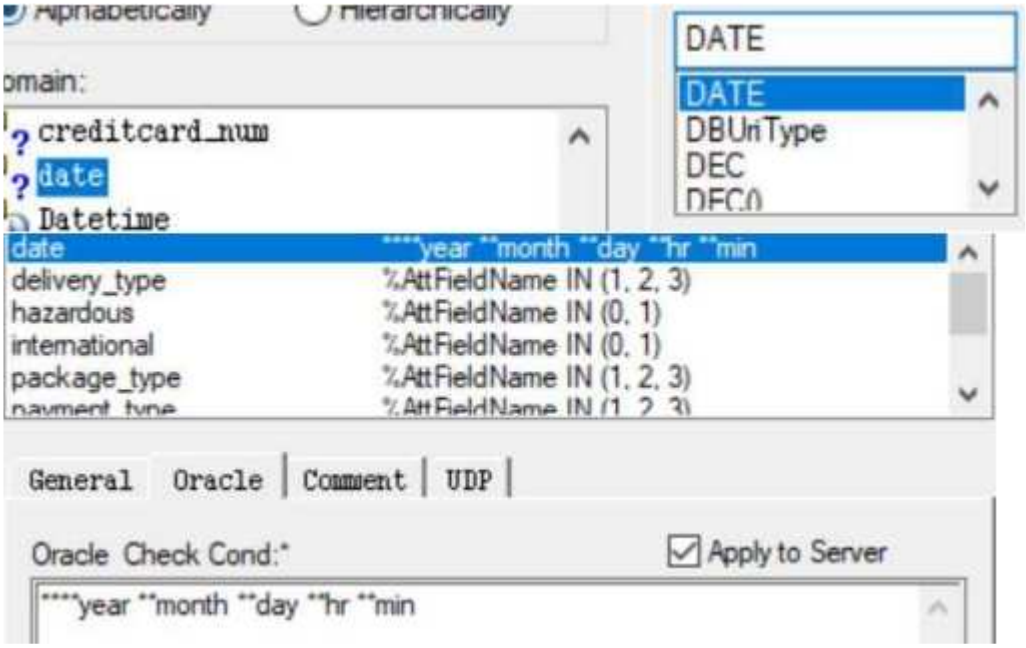
5-4). city

도시를 의미하는 city의 경우 구글에서 찾아본 결과 가장 긴 도시이름의 길이가 200을 넘지 않았으므로 CHAR(200)으로 지정해주었다. 일하는 곳의 도시이름이 없을수 없으므로 NOT NULL로 해주었다.

5-5). creditcard_num

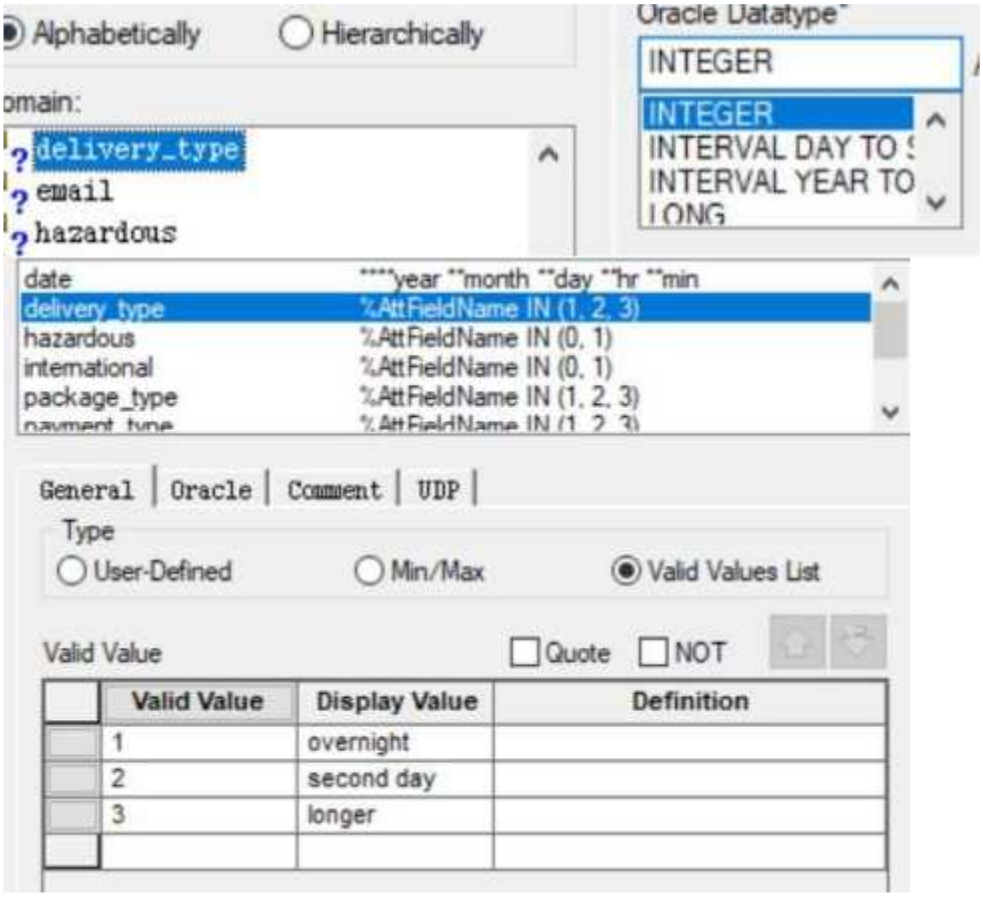
카드번호를 의미하는 creditcard_num의 경우 국제적으로 사용되는 Visa, MasterCard와 같은 카드들은 모두 16자리의 숫자이므로 CHAR(16)으로 지정해주었다. 카드결제방식이 아닌 다른 결제방식으로 사용할 수 있으므로 NULL 값을 허용해 주었다.

5-6). date



년, 월, 일, 시간 등을 의미하는 date의 경우에는 validation rule을 적용하여 "****year **month **day **hr **min"으로 정의를 해주었다.

5-7) delivery_type

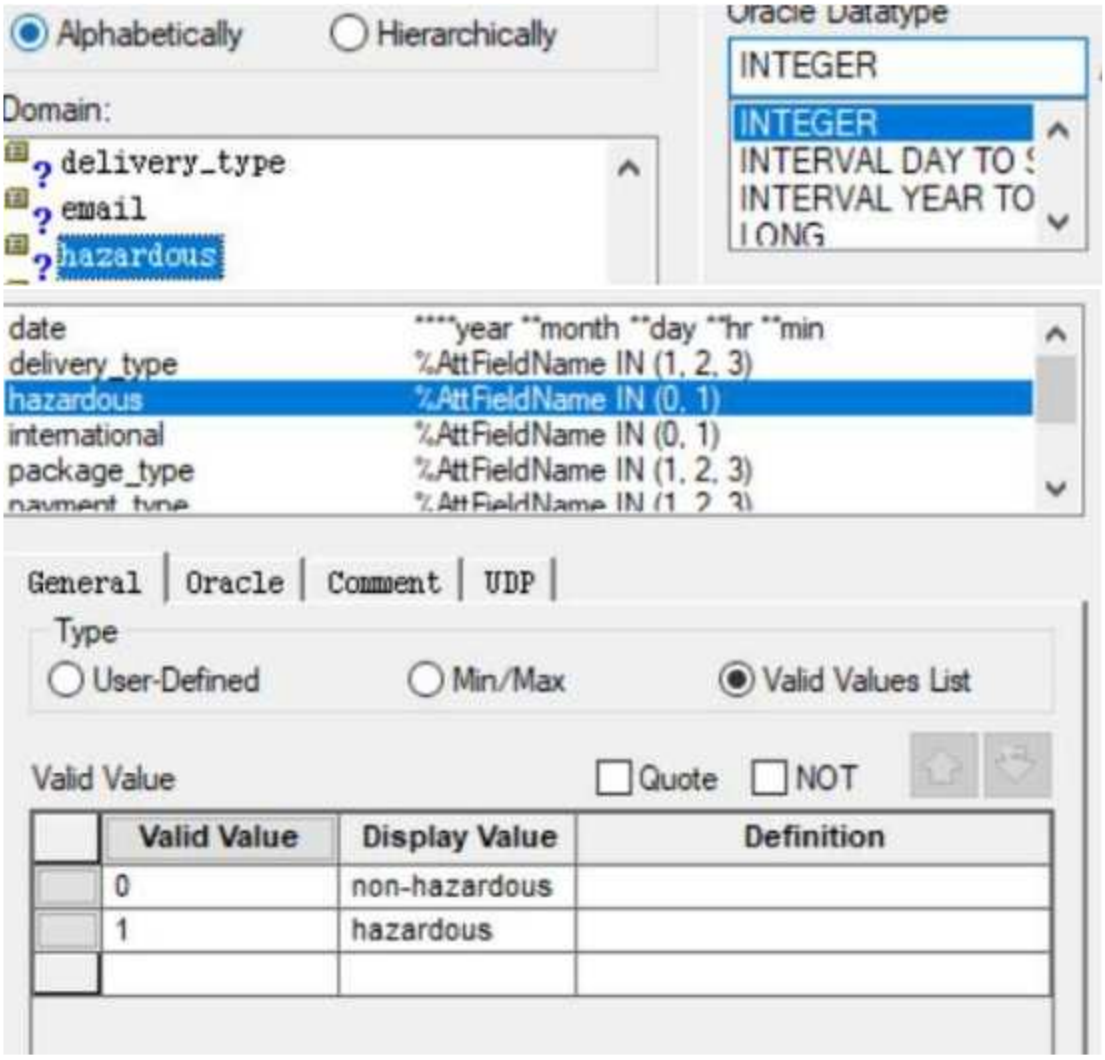


배송예정시간을 의미하는 delivery_time의 경우에는 INTERGER typed으로 선언해주었는데 validation rule을 적용하여 1의 값을 가질 경우 overnight, 2의 값을 가질 경우 second day, 3의 값을 가질 경우 longer를 의미하게 해주었다. 3가지 type중 무조건 한 가지를 만족해야하므로 NOT NULL로 해주었다.

5-8). email

email의 경우 보통 30자를 넘어가지 않으므로 CHAR(30)으로 선언해주었고, email주소가 없어도 배송에는 지장이 없으므로 NULL값을 허용해 주었다.

5-9). hazardous

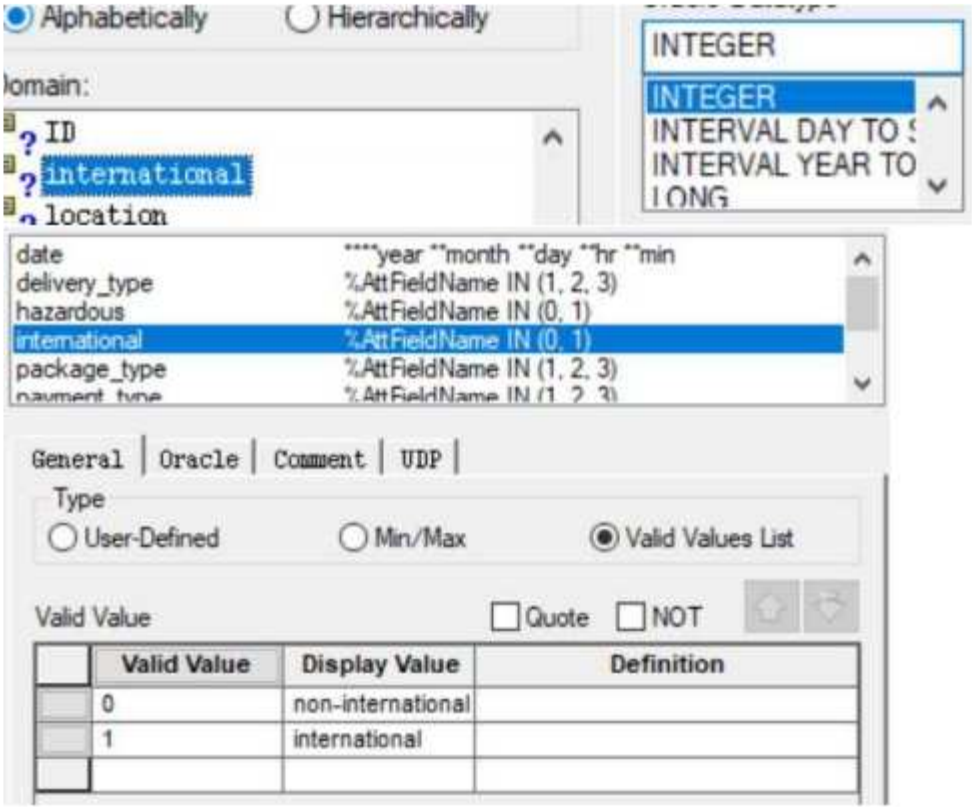


위험물을 뜻하는 hazardous의 경우 0또는 1의 값을 가지는 INTERGER type으로 선언해주었다. 0은 위험물이 아님을 뜻하는 non-hazardous, 1은 위험물을 뜻하는 hazardous를 의미하게 해주었다. package의 hazardous여부는 non-hazardous혹은 hazardous둘 중 하나의 값을 무조건 가지게 되므로 NOT NULL로 해주었다.

5-10). ID

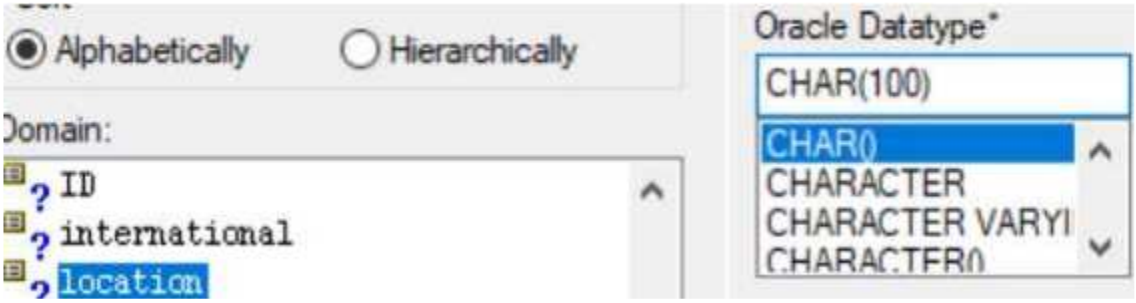
소비자, 배송원, 운송수단 등을 name을 가지고는 unique하게 구별해 낼 수 없으므로 ID값을 만들어주었고, 최대 50길이의 CHAR type으로 선언해주었다. ID는 구별요소이기 때문에 무조건적으로 가져야하므로 NOT NULL로 해주었다 NOT NULL의 예외로는 customer entity에 존재하는 shipper_ID의 경우 배송부와 손님간의 계약이 존재하지 않을 수 있기 때문에 NULL값을 허용해주었다.

5-11). international



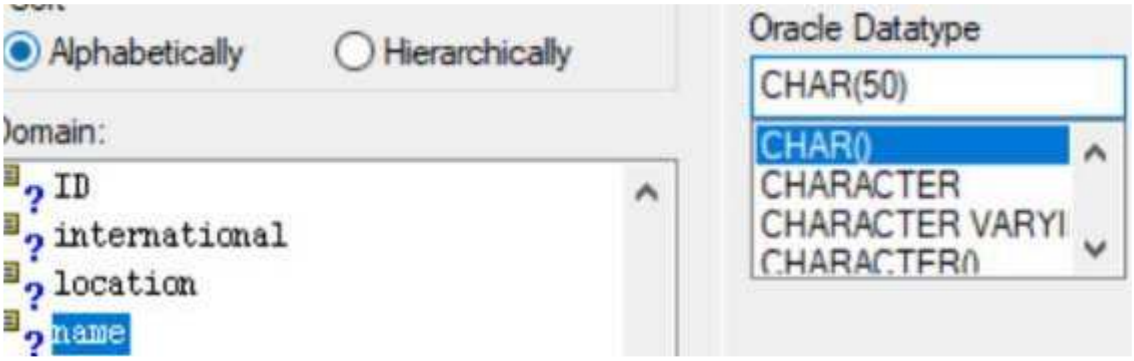
해외배송물을 뜻하는 international의 경우 0또는 1의 값을 가지는 INTERGER type으로 선언해주었다. 0은 국내배송을 뜻하는 non-international, 1은 해외배송물을 뜻하는 international을 의미하게 해주었다. package의 international여부는 non-international혹은 international 둘 중 하나의 값을 무조건 가지게 되므로 NOT NULL로 해주었다.

5-12). location



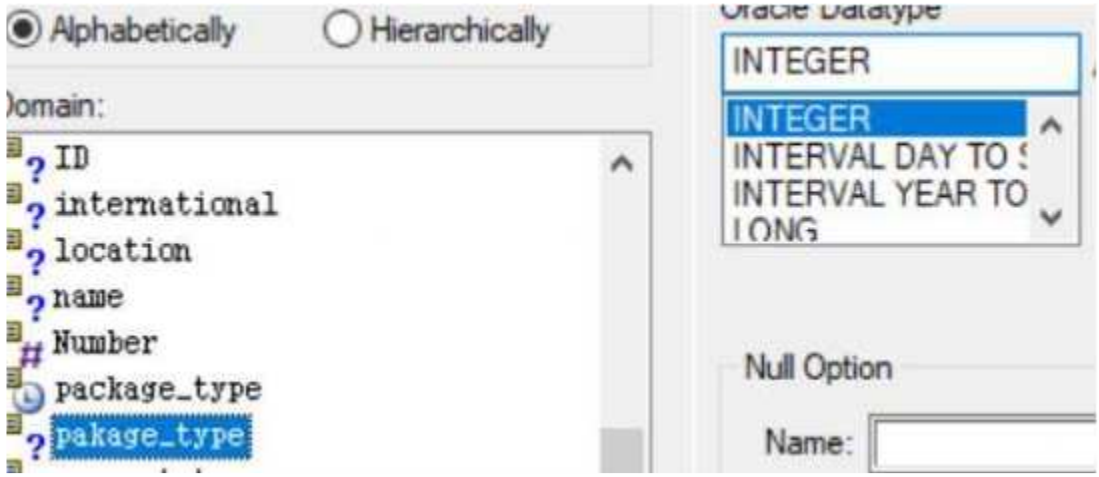
현재의 장소, 위치 등을 표현하는 location의 경우는 CHAR(100)을 통해 길이를 100으로 제한해주었다. 또한 위치정보는 무조건적으로 가지고 있어야 하는 값이므로 NOT NULL로 해주었다.

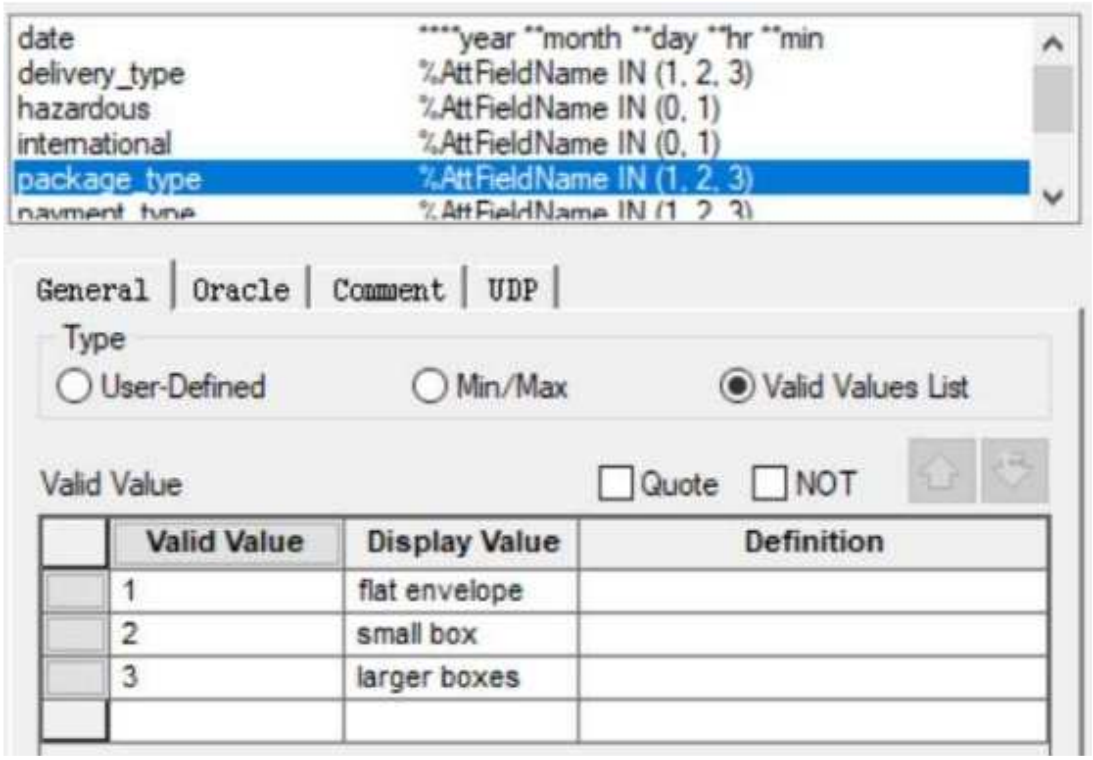
5-13). name



이름을 의미하는 name의 경우 영어이름만을 받을 수 있고 길이는 50을 넘지 않게 제한하여 CHAR(50)으로 선언해주었다. 물건 배송시에 이름정보는 발송인과 수령인 모두에게 요구되므로 NOT NULL로 해주었다.

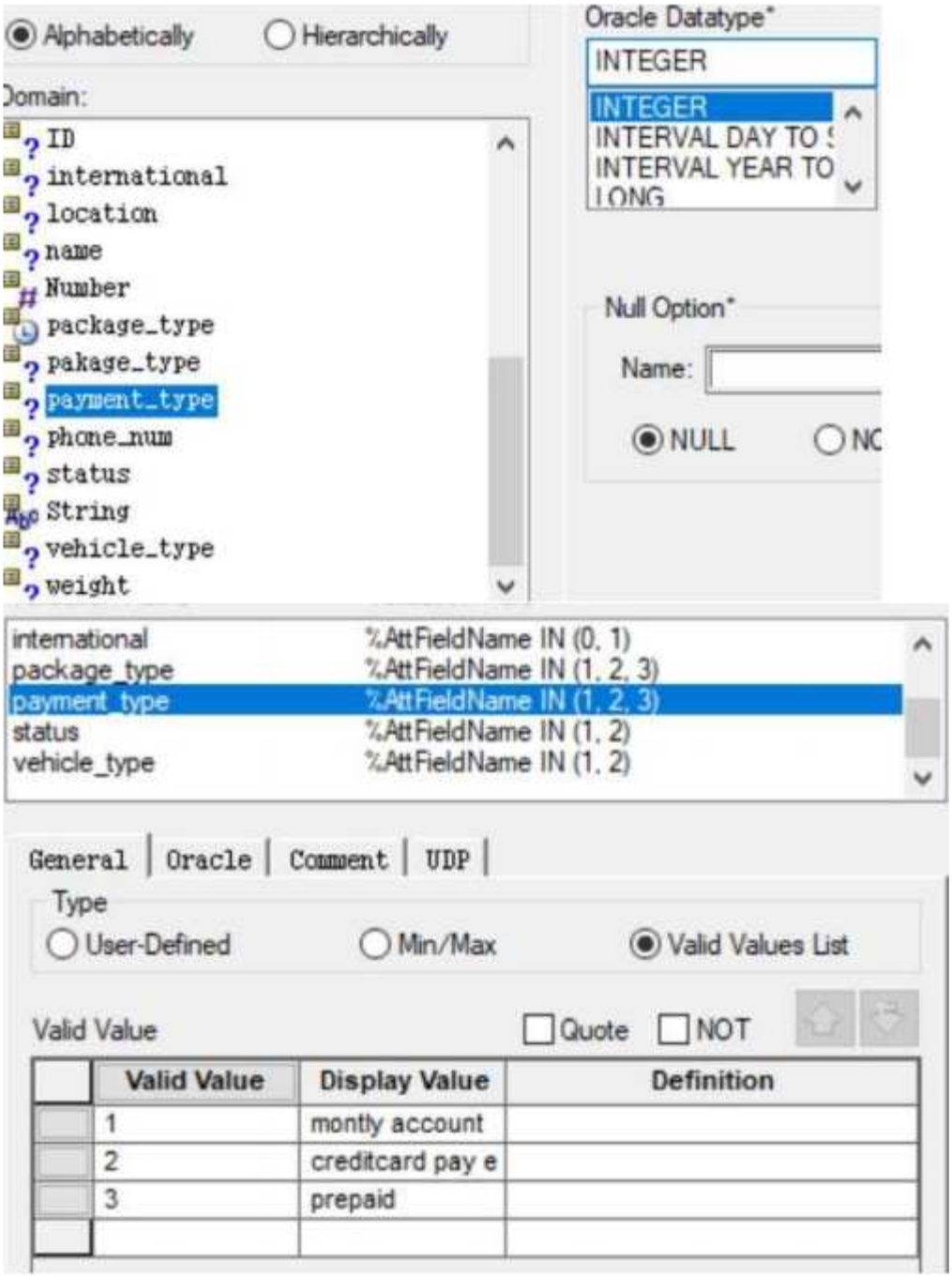
5-14). package_type





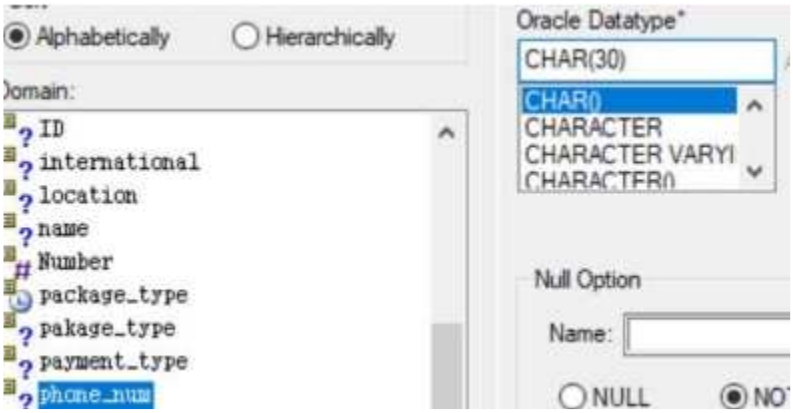
package의 포장형태를 의미하는 package_type의 경우는 validation rule을 이용하여 1, 2, 3의 값을 가질 수 있는 INTERGER type의로 선언해주었다. 각각 값을 살펴보면, 1의 값을 가지게 되면 flat envelope, 2의 값을 가지게 되면 small box, 3의 값을 가지게 되면 larger boxes를 의마하게 하였다. 배송품의 포장형태는 무조건 적으로 위의 값들 중 하나의 값을 가져야 하기 때문에 NOT NULL로 해주었다.

5-15). payment_type



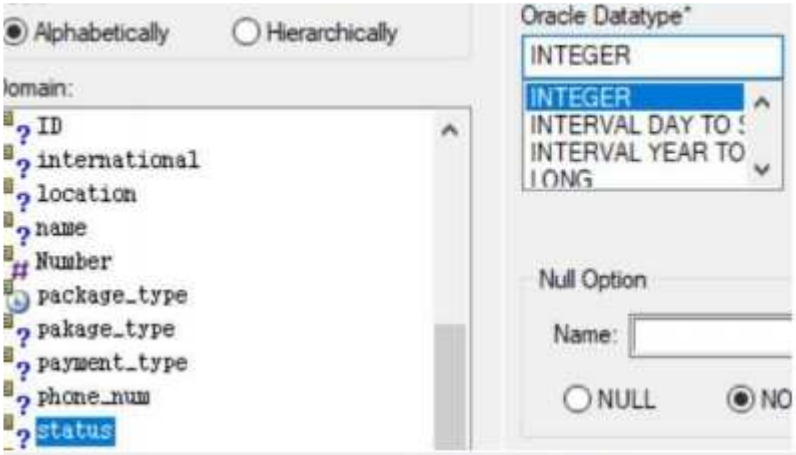
결제방식을 의마하는 payment_type의 경우는 validation rule을 이용하여 1, 2, 3의 값을 가질 수 있는 INTEGER type으로 선언해주었다. 각각 값을 살펴보면, 1의 값을 가지게 되면 montly_account(달마다 account로 pay), 2의 값을 가지게 되면 creditcard_pay_each(배송시킬때마다 카드결제), 3의 값을 가지게 되면 prepaid(선결제)를 의미하게 하였다. 배송의 결제방식은 무조건적으로 위의 값들중 하나의 값을 가져야 되기 때문에 NOT NULL로 해주었다.

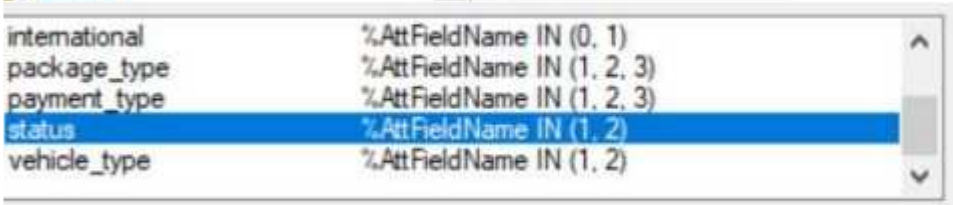
5-16). phone_num

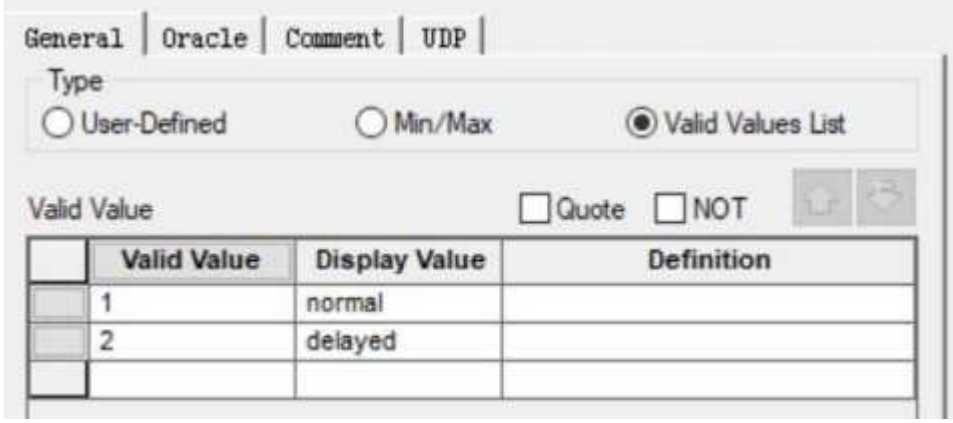


전화번호를 의미하는 phone_num의 경우 전 세계적으로 전화번호가 30자를 넘지 않기 때문에 30자로 길이를 제한하였고 배송 진행시에 수령인과 발송인 모두의 전화번호는 필수적이므로 NOT NULL로 해주었다.

5-17). track_status, Destroyed



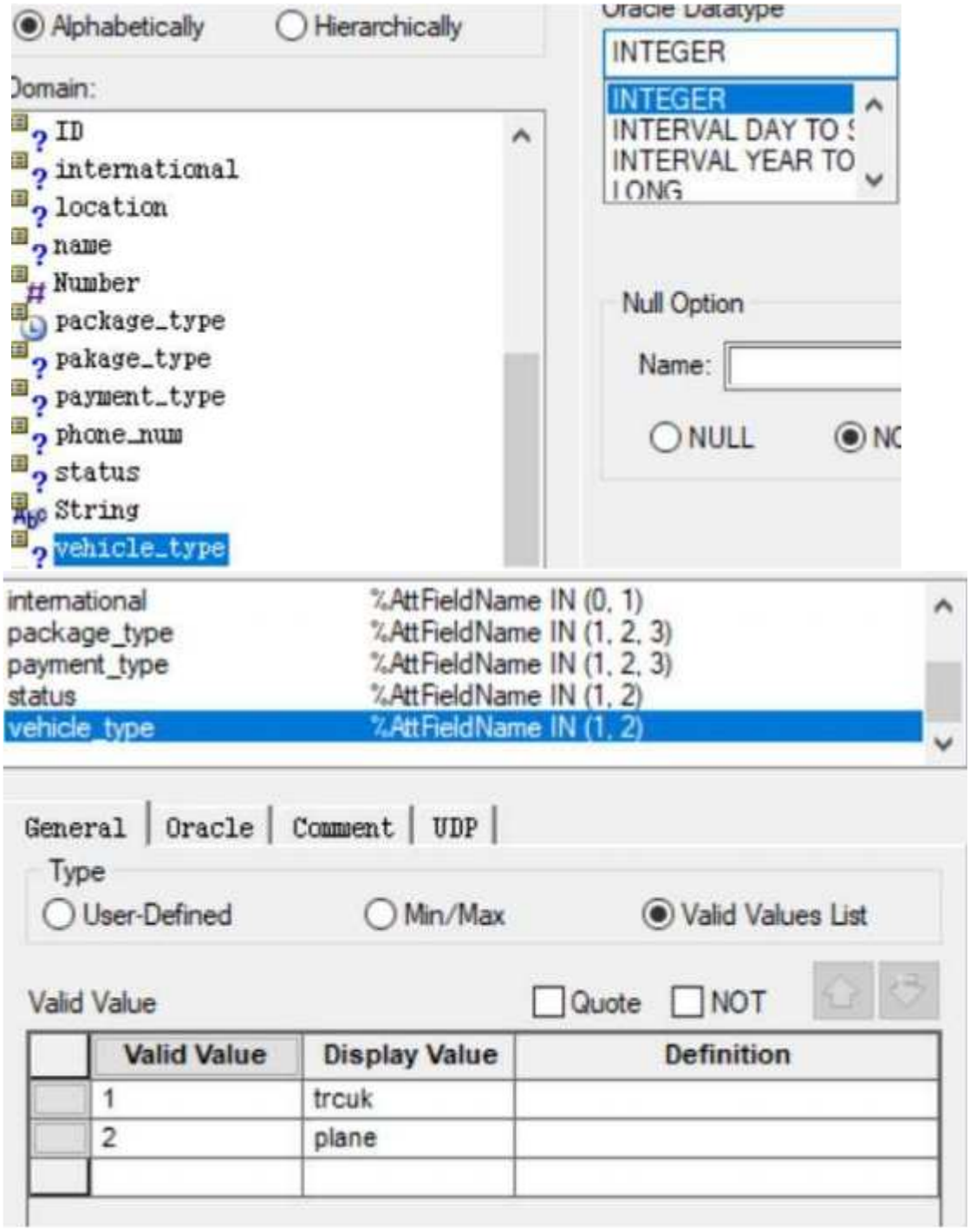




현재 배송상태를 의미하는 status의 경우는 validation rule을 이용하여 1, 2의 값을 가질 수 있는 INTERGER type으로 선언해주었다. 각각 값을 살펴보면, 1의 값을 가지게 되면 normal(정상배송상태), 2의 값을 가지게 되면 delayed(배송지연상태[ex)truck crash])를 의미하게 하였다. 배송은 항상 정상배송 아니면 지연배송이므로 NOT NULL로 해주었다.

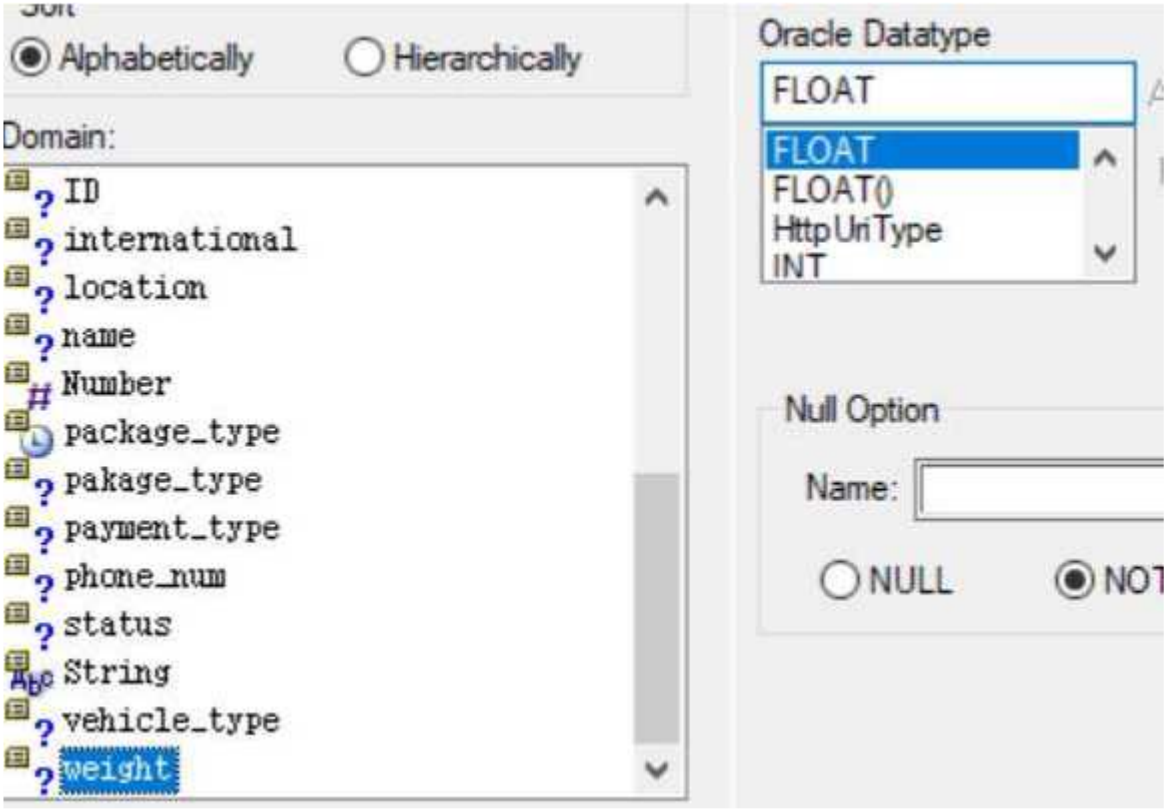
또한 Destroyed도 일종의 delay상태라 볼 수 있으므로 status type으로 선언해주었다.

5-18). vehicle_type



현재 운송중인 기계의 종류를 의미하는 vehicle_type의 경우는 validation rule 을 이용하여 1, 2의 값을 가질 수 있는 INTEGER type으로 선언해주었다. 각각 값을 살펴보면, 1의 값을 가지게 되면 truck(트럭운송), 2의 값을 가지게 되면 plane(항공배송)을 의미하게 하였다. 배송진행은 무조건 트럭 혹은 비행기를 통해 된다하였으므로 NOT NULL로 해주었다.

5-19). weight



물건의 무게를 의미하는 weight의 경우에는 무게는 2.19kg처럼 소수점을 가지게 되므로 FLOAT형태로 선언해주었다. 또한 모든 배송품은 무조건적으로 무게를 가지므로 NOT NULL로 해주었다.

마지막으로 bill에 존재하는 purchase_date는 정보를 쉽게 얻어내기 위하여 purchase_year, purchase_month, purchase_day로 나눠 주었으며 INTEGER type으로 선언해주었다(년, 원, 일은 모두 숫자이므로). 그러나 고객에게 정확한 배송추적을 제공하기위해 location의 date는 TIMESTAMP형식으로 선언해 정확한 시간(sec단위까지)을 제공하게 하였다.

3. Queries

a). query I-1

```
select customer_name
from customer, package, package_vehicle
where vehicle_ID = 'TR-1721' and package.package_ID = package_vehicle.package_ID and customer.customer_ID = package.customer_ID
```

쿼리 1-1의 경우 1721번의 트럭이 파손되었을 경우 트럭안에 존재하는 모든 택배의 발송자를 찾아내는 구문으로 우선적으로 vehicle_ID = 'TR-1721'을 통해 트럭을 찾고, package.packge_ID = package_vehicle.package_ID를 통해 데이터의 일관성을 유지한후에 customer.customer_ID = package_customer_ID를 통해 중복된 정보가 제거된 customer_name을 찾아내었다.

b). query I-2

```
select recipient_name
from customer, package, package_vehicle
where vehicle_ID = 'TR-1721' and package.package_ID = package_vehicle.package_ID and customer.customer_ID = package.customer_ID
```

쿼리 1-2의 경우 1721번의 트럭이 파손되었을 경우 트럭안에 존재하는 모든 택배의 수신자 이름을 찾아내는 구문으로 우선적으로 vehicle_ID = 'TR-1721'을 통해 트럭을 찾고, package.packge_ID = package_vehicle.package_ID를 통해 데이터의 일관성을 유지한후에 customer.customer_ID = package.customer_ID를 통해 중복된 정보를 제외한뒤에 package table에서 택배수신자의 이름인 recipient_name을 뽑아주었다.

c). query I-3

```
select distinct location_name
from package, package_vehicle, location
where vehicle_ID = 'TR-1721' and package.package_ID = package_vehicle.package_ID and location.package_ID = package.package_ID and destroyed = '2'
```

쿼리 1-3의 경우 1721번의 트럭이 destroyed되기 이전까지 도착한 장소를 찾아내는 구문으로 우선적으로 vehicle_ID = 'TR-1721'을 통해 트럭을 찾고, package.packge_ID = package_vehicle.package_ID와 location.package_ID = package.package_ID를 통해 데이터의 일관성을 유지한채로 만들어두었던 파괴상태를 의미하는 destroyed attribute를 통해 파괴직전 출발장소인 location_name을 출력해주었다.

d). query II

```
select customer.customer_name
from ( select max(P.dal) as del from (select count(package_ID) as dal from (select * from bill where bill.purchase_year = '2016') as N group by customer_ID) as P) as R
, (select count(package_ID) as S, customer_ID from (select * from bill where bill.purchase_year = '2016') as M group by customer_ID) as B, customer
where B.S = R.del and B.customer_ID = customer.customer_ID
```

쿼리 2의 경우 사용자의 입력을 받아 그 해당년도에 가장 주문을 많이한 소비자를 찾아내는 구문으로 쿼리가 복잡하므로 단계별로 살펴보자.

d-1). (select max(P.dal) as del from (select count(package_ID) as dal from (select * from bill where bill.purchase_year = '****') as N group by customer_ID) as P) as R <-1번째 relation

위의 색깔은 쿼리 각각의 영역을 나타낸 것이다. 쿼리를 해석해보자면 먼저 bill table을 사용자가 지정한 년도별로 정리한 table을 만들어 N이라 이름 지었다. 이후 그러한 테이블에서 customer_ID별로 grouping을 해주고 count를 통해 각 customer가 몇 개의 택배를 주문했는지와

customer_ID를 포함한 table을 만들었다. 이후에 max구문을 이용하여 가장 많이 택배를 주문한 개수만을 가지는 table을 만들어 주었다.

d-2). (select count(package_ID) as S, customer_ID from (select * from bill where bill.purchase_year = '****') as M group by customer_ID) as B, customer <-2번째 relation
위에서 구한 max개수를 이용해서 customer_ID를 찾기 위해 bill table을 이용하여 사용자가 입력한 년도의 값을 가지며 customer_ID와 택배의 개수를 가지는 table을 만들어주었다. 이후 앞의 table과 현재의 table, 그리고 customer를 합쳐준 뒤에 where B.S = R.del and B.customer_ID = customer.customer_ID 구문을 이용하여 구한 최대의 택배개수를 주문한 소비자의 이름을 뽑아냈다. 위에서 ****로 표기된 부분만 변경해주면 원하는 년도의 값을 얻을 수 있다. 본인의 Database에서 Query에 사용가능한 년도는 2016, 2017, 2018년 이다.

e). query III

```
select customer_name
from (select max(charge) as pal from bill, customer where customer.customer_ID = bill.customer_ID and bill.purchase_year = '2016') as S, bill, customer
where S.pal = bill.charge and bill.customer_ID = customer.customer_ID
```

쿼리 3의 경우 지정한 년도에 가장 많은 금액을 사용한 소비자를 찾아내는 구문으로 우선적으로 nested query인 select max(charge) as pal from bill, customer where customer.customer_ID = bill.customer_ID and bill.purchase_year = '****'부터 살펴보자.
위의 구문은 우선적으로 customer.customer_ID = bill.customer_ID를 통해 join후에 데이터의 일관성을 유지하였고, 이후 bill.purchase_year = '****'를 통해 사용자가 원하는 년도의 가장 많은 금액인 max(charge)를 알 수 있다. 이후 위에서 얻어진 max(charge)값만을 가지는 relation S와 bill, 그리고 customer를 join해준뒤 where S.pal = bill.charge and bill.customer_ID = customer.customer_ID를 통해 그 가장큰 금액을 어떤사람이 썼는지 customer_name을 뽑아내었다. 위에서 ****로 표기된 부분만 변경해주면 원하는 년도의 값을 얻을 수 있다. 본인의 Database에서 Query에 사용가능한 년도는 2016, 2017, 2018년 이다.

f). query VI

```
select package_ID
from tracking
where track_status = 2
```

쿼리 4의 경우 배송이 지연된 배송품의 송장번호를 찾아내는 구문으로 er-model을 설계할 때 배송상태(1:정상배송, 2:배송지연)를 attribute로서 만들어주었기 때문에 단순히 track_status = 2를 통해 배송이 지연된 송장번호를 바로 얻을 수 있었다.

g). query V

```
select customer_name, bill.address, bill.charge, package_type, weight, delivery_type
from bill, customer
where customer_name = 'Saram' and bill.purchase_year = '2018' and bill.purchase_month = '03'
and bill.customer_ID = customer.customer_ID
```

쿼리 5의 경우 특정 소비자의 영수증을 만들어주는 구문으로 위의 cutomer_name, purchase_year,

purchase_month를 사용자에게 입력받아 그에 맞는 소비자의 영수증을 파일로 만들어준다. 이를 C언어 파일입출력으로 구현하기 이전에 다음과 같은 쿼리를 통해 원하는 정보를 확인할 수 있다.

4. Code implementation

4-1. Destroyed truck

```
**********************************************
** Sogang Package Delivery Management System **
**********************************************


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

        1. TYPE I
        2. TYPE II
        3. TYPE III
        4. TYPE IV
        5. TYPE V
        0. QUIT
Which type of query? 1

---- TYPE I ----
Input the number of truck : 1366
Truck 1366 is not destroyed

Input the number of truck : 1721

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

---- TYPE I-1 ----
** Find all customers who had a package on the truck at the time of the crash. **
Customer Name : Bandel. Thuk.
```

1721번 트럭이 destroyed되었다 명시되어 있으므로 파괴된 트럭의 번호가 들어오기 전까지 트럭의 번호를 계속 입력받는다.

4-2. TYPE I

파손된 트럭의 입력을 받은 이후 사용자의 입력(1, 2, 3)에 따라 파손된 트럭의 배송물을 보낸사람(TYPE I-1), 배송물 수령인(TYPE I-2), 파손이전 마지막 배송지인(TYPE I-3)을 출력해준다. 먼저 데이터를 살펴보면 다음과 같다.

```
insert into package_vehicle values('t-201701', '2016-02-15 17:20:00', 'TR-1721');
insert into package_vehicle values('t-201702', '2016-02-16 12:30:00', 'TR-1721');
```

<특정 배송물의 트럭의 정보>

```
insert into package values('t-201701', 'c-20103177', '1', '0', 'Arnold', '1930', 'Arizona');
insert into package values('t-201702', 'c-20104231', '0', '1', 'Klop', '1931', 'Kensas');
```

<특정 배송물의 발송인과 수령인>

```
insert into location values('t-201701', '2016-02-15 17:20:00', 'Mecigan', 'Colorado', '1');
insert into location values('t-201701', '2016-02-16 17:20:00', 'Colorado', 'Texas', '1');
insert into location values('t-201701', '2016-02-17 17:20:00', 'Texas', 'New york', '2');
```

<특정 배송물의 현재위치, 목표지점 정보 마지막 attribute가 1이면 파손x, 2면 파손>

즉 위와 같이 먼저 트럭 1721번에는 t-201701택배와 t-201702택배가 들어가 있고, t-201701택배의 발송인은 c-20103177 (Bandel)이고 수령인은 Arnold이며, t-201702택배의 발송인은 c-20104231(Thuk)이고 수령인은 Klop이다. 또한 경로를 보면 파손전 마지막 위치는 Texas인 것을 알 수 있다.

그리하여 I-1번을 수행하면 Bandel, Thuk의 이름이 나와야하고, I-2번을 수행하면 Arnold, Klop의 이름이 나와야 하며 마지막으로 I-3번을 수행하면 Texas가 출력이 되야 한다. 프로그램의 출력 결과는 다음과 같다.

```
---- TYPE I-1 ----
** Find all customers who had a package on the truck at the time of the crash. **
Customer Name : Bandel. Thuk.

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

---- TYPE I-2 ----
** Find all recipients who had a package on the truck at the time of the crash. **
Recipient Name : Arnold. Klop.

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

---- TYPE I-3 ----
** Find the last successful delivery by that truck prior to the crash. **
Last successful delivery : Texas
```

결과가 정상적으로 잘 나옴을 알 수 있다.
또한 아래와 같이 0을 입력할시에 정상적으로 종료된다.

```
---- TYPE I-2 ----
** Find all recipients who had a package on the truck at the time of the crash. **
Recipient Name : Arnold. Klop.

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

---- TYPE I-3 ----
** Find the last successful delivery by that truck prior to the crash. **
Last successful delivery : Texas

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

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

        1. TYPE I
        2. TYPE II
        3. TYPE III
        4. TYPE IV
        5. TYPE V
        0. QUIT
Which type of query?
```

4-3. TYPE II

메뉴 II는 사용자의 년도 입력을 받아 그 년도에 가장 많은 배송을 한 소비자를 찾아내는 메뉴이다. 우선적으로 사용자의 패키지 주문 데이터를 bill table에 다음과 같이 insert해주었다.

```
insert into bill values('t-201701', '2.31', '2016', '02', '14', '1',  '3.14', '3', 'Miami', 'c-20104231');
insert into bill values('t-201722', '2.40', '2016', '02', '17', '2',  '4.02', '2', 'Florida', 'c-20104231');
insert into bill values('t-201702', '6.12', '2016', '03', '21', '3',  '5.22', '1', 'Kensas', 'c-20106832');
insert into bill values('t-201703', '3.25', '2016', '04', '03', '1',  '2.87', '2', 'Texas', 'c-20103177');
insert into bill values('t-201704', '5.12', '2016', '05', '08', '2',  '3.62', '2', 'Florida', 'c-20102372');
insert into bill values('t-201705', '3.78', '2016', '06', '28', '2',  '2.91', '1', 'California', 'c-20104993');
insert into bill values('t-201706', '2.12', '2017', '01', '11', '1',  '3.67', '1', 'New york', 'c-20102819');
insert into bill values('t-201707', '3.02', '2017', '02', '16', '3',  '6.17', '2', 'Colorado', 'c-20104339');
insert into bill values('t-201708', '1.99', '2017', '02', '19', '1',  '4.66', '3', 'Utah', 'c-20103892');
insert into bill values('t-201721', '2.13', '2017', '02', '20', '2',  '5.13', '2', 'Ohio', 'c-20103892');
insert into bill values('t-201709', '2.38', '2017', '03', '05', '1',  '3.82', '3', 'Arizona', 'c-20107261');
insert into bill values('t-201710', '4.08', '2017', '07', '23', '2',  '2.16', '3', 'Oregon', 'c-20102221');
insert into bill values('t-201711', '4.12', '2018', '02', '14', '1',  '4.73', '1', 'Ohio', 'c-20108132');
insert into bill values('t-201712', '2.76', '2018', '02', '26', '3',  '4.23', '2', 'Seoul', 'c-20101932');
insert into bill values('t-201713', '3.18', '2018', '03', '18', '3',  '6.31', '2', 'Busan', 'c-20108118');
insert into bill values('t-201714', '7.14', '2018', '03', '29', '1',  '1.23', '1', 'Daegu', 'c-20103013');
insert into bill values('t-201715', '1.73', '2018', '04', '23', '2',  '2.32', '1', 'Daezeon', 'c-20106912');
insert into bill values('t-201720', '2.89', '2018', '02', '27', '3',  '4.1', '2', 'Seoul', 'c-20108118');
```

즉, 위에서 보면 2016년에는 c-20104231의 ID를 가진 사람이 가장 많은 주문을, 2017년에는 c-20103892의 ID를 가진 사람이 가장 많은 주문을, 마지막으로 2018년에는 c-20108118의 ID를 가진 사람이 가장 많은 주문을 진행하였다. 그러므로 customer table의 정보를 이용하여 각 년도의 최다 배송자를 확인해보면 2016년도는 Thuk, 2017년도는 Snow, 2018년도는 Minseok이다.

출력의 결과는 다음과 같다.

```
---- TYPE II ----
** Find the customer who has shipped the most packages in certain year **
which year? 2016
Customer Name : Thuk
** Find the customer who has shipped the most packages in certain year **
which year? 2017
Customer Name : Snow
** Find the customer who has shipped the most packages in certain year **
which year? 2018
Customer Name : Minseok
** Find the customer who has shipped the most packages in certain year **
which year? 2019
There are no purchase in year 2019! Please enter year (2016~2018)
** Find the customer who has shipped the most packages in certain year **
which year? 0

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

        1. TYPE I
        2. TYPE II
        3. TYPE III
        4. TYPE IV
        5. TYPE V
        0. QUIT
Which type of query?
```

2016~2018년도의 입력만 받으며, 0을 받으면 종료되고 각 년도에 맞는 이름값이 출력되므로 정상적으로 수행됨을 알 수 있다.

4-3. TYPE III

메뉴 III는 사용자의 년도 입력을 받아 그 년도에 배송에 가장 많은 돈을 소비한 소비자를 찾아내는 메뉴이다. 우선적으로 사용자의 패키지 주문 데이터를 bill table에 다음과 같이 insert해주었다.

```
insert into bill values('t-201701', '2.31', '2016', '02', '14', '1',  '3.14', '3', 'Miami', 'c-20104231');
insert into bill values('t-201722', '2.40', '2016', '02', '17', '2',  '4.02', '2', 'Florida', 'c-20104231');
insert into bill values('t-201702', '6.12', '2016', '03', '21', '3',  '5.22', '1', 'Kensas', 'c-20106832');
insert into bill values('t-201703', '3.25', '2016', '04', '03', '1',  '2.87', '2', 'Texas', 'c-20103177');
insert into bill values('t-201704', '5.12', '2016', '05', '08', '2',  '3.62', '2', 'Florida', 'c-20102372');
insert into bill values('t-201705', '3.78', '2016', '06', '28', '2',  '2.91', '1', 'California', 'c-20104993');
insert into bill values('t-201706', '2.12', '2017', '01', '11', '1',  '3.67', '1', 'New york', 'c-20102819');
insert into bill values('t-201707', '3.02', '2017', '02', '16', '3',  '6.17', '2', 'Colorado', 'c-20104339');
insert into bill values('t-201708', '1.99', '2017', '02', '19', '1',  '4.66', '3', 'Utah', 'c-20103892');
insert into bill values('t-201721', '2.13', '2017', '02', '20', '2',  '5.13', '2', 'Ohio', 'c-20103892');
insert into bill values('t-201709', '2.38', '2017', '03', '05', '1',  '3.82', '3', 'Arizona', 'c-20107261');
insert into bill values('t-201710', '4.08', '2017', '07', '23', '2',  '2.16', '3', 'Oregon', 'c-20102221');
insert into bill values('t-201711', '4.12', '2018', '02', '14', '1',  '4.73', '1', 'Ohio', 'c-20108132');
insert into bill values('t-201712', '2.76', '2018', '02', '26', '3',  '4.23', '2', 'Seoul', 'c-20101932');
insert into bill values('t-201713', '3.18', '2018', '03', '18', '3',  '6.31', '2', 'Busan', 'c-20108118');
insert into bill values('t-201714', '7.14', '2018', '03', '29', '1',  '1.23', '1', 'Daegu', 'c-20103013');
insert into bill values('t-201715', '1.73', '2018', '04', '23', '2',  '2.32', '1', 'Daezeon', 'c-20106912');
insert into bill values('t-201720', '2.89', '2018', '02', '27', '3',  '4.1', '2', 'Seoul', 'c-20108118');
```

즉, 위에서 보면 2016년에는 c-2010683의 ID를 가진 사람이 가장 많은 소비를, 2017년에는 c-20102221의 ID를 가진 사람이 가장 많은 소비를, 마지막으로 2018년에는 c-20103013의 ID를 가진 사람이 가장 많은 소비를 진행하였다. 그러므로 customer table의 정보를 이용하여 각 년도의 최다 배송자를 확인해보면 2016년도는 Joen, 2017년도는 Chilwell, 2018년도는 Saram이다.

출력의 결과는 다음과 같다.

```
---- TYPE III ----
** Find the customer who has spent the most money on shipping in the past certain year **
which year? 2016
Customer Name : Joen
** Find the customer who has spent the most money on shipping in the past certain year **
which year? 2017
Customer Name : Chilwell
** Find the customer who has spent the most money on shipping in the past certain year **
which year? 2018
Customer Name : Saram
** Find the customer who has spent the most money on shipping in the past certain year **
which year? 2019
There are no purchase in year 2019! Please enter year (2016~2018)
** Find the customer who has spent the most money on shipping in the past certain year **
which year? 0

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

        1. TYPE I
        2. TYPE II
        3. TYPE III
        4. TYPE IV
        5. TYPE V
        0. QUIT
Which type of query?
```

2016~2018년도의 입력만 받으며, 0을 받으면 종료되고 각 년도에 맞는 이름값이 출력되므로 정상적으로 수행됨을 알 수 있다.

4-4. TYPE IV

메뉴 IV는 배송이 지연된 품목을 출력해주는 메뉴로서 본인의 database에서는 tracking entity에 track_status라는 attribute를 만들어 줌으로서 빠르게 확인할 수 있었다. 우선적으로 insert구문을 보면 다음과 같다.

```
insert into tracking values('t-201701', '2', 'Mike');
insert into tracking values('t-201702', '1', 'Thomas');
insert into tracking values('t-201703', '1', 'Ben');
insert into tracking values('t-201704', '1', 'Ruby');
insert into tracking values('t-201705', '2', 'Big');
insert into tracking values('t-201706', '1', 'Martin');
insert into tracking values('t-201707', '1', 'Sam');
insert into tracking values('t-201708', '1', 'Jhon');
insert into tracking values('t-201709', '1', 'Aria');
insert into tracking values('t-201710', '1', 'Sombra');
insert into tracking values('t-201711', '2', 'Piter');
insert into tracking values('t-201712', '1', 'Minsik');
insert into tracking values('t-201713', '1', 'Heungjun');
insert into tracking values('t-201714', '2', 'Yongdam');
insert into tracking values('t-201715', '1', 'Donghyuk');
insert into tracking values('t-201720', '1', 'Sami');
insert into tracking values('t-201721', '1', 'Samsung');
insert into tracking values('t-201722', '1', 'LG');
```

가운데 attribute를 확인해보면 1이 정상배송 2가 지연배송이므로 메뉴에서 출력을 하게 되면 t-201701, t-201705, t-201711, t-201714가 출력으로 나와야 한다. 프로그램의 출력결과는 다음과 같다.

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

        1. TYPE I
        2. TYPE II
        3. TYPE III
        4. TYPE IV
        5. TYPE V
        0. QUIT
Which type of query? 4

---- TYPE IV ----
** Find those packages that were not delivered within the promised time **
t-201701, t-201705, t-201711, t-201714,
```

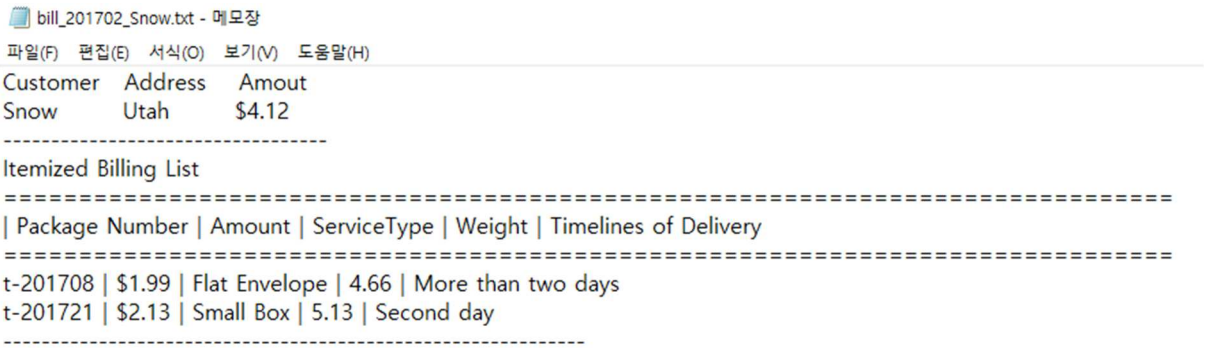
기대했던 값이 출력이 되므로 정상적으로 수행됨을 알 수 있다.

4-5 TYPE V

메뉴 V는 사용자에게 입력을 사람의 이름과 년도/월 을 입력받아 입력받은 사람에게 해당하는 년도의 영수증을 만들어주는 구문이다. 우선적으로 영수증 table은 다음과 같다.

```
insert into bill values('t-201701', '2.31', '2016', '02', '14', '1',  '3.14', '3', 'Miami', 'c-20104231');
insert into bill values('t-201722', '2.40', '2016', '02', '17', '2',  '4.02', '2', 'Florida', 'c-20104231');
insert into bill values('t-201702', '6.12', '2016', '03', '21', '3',  '5.22', '1', 'Kensas', 'c-20106832');
insert into bill values('t-201703', '3.25', '2016', '04', '03', '1',  '2.87', '2', 'Texas', 'c-20103177');
insert into bill values('t-201704', '5.12', '2016', '05', '08', '2',  '3.62', '2', 'Florida', 'c-20102372');
insert into bill values('t-201705', '3.78', '2016', '06', '28', '2',  '2.91', '1', 'California', 'c-20104993');
insert into bill values('t-201706', '2.12', '2017', '01', '11', '1',  '3.67', '1', 'New york', 'c-20102819');
insert into bill values('t-201707', '3.02', '2017', '02', '16', '3',  '6.17', '2', 'Colorado', 'c-20104339');
insert into bill values('t-201708', '1.99', '2017', '02', '19', '1',  '4.66', '3', 'Utah', 'c-20103892');
insert into bill values('t-201721', '2.13', '2017', '02', '20', '2',  '5.13', '2', 'Ohio', 'c-20103892');
insert into bill values('t-201709', '2.38', '2017', '03', '05', '1',  '3.82', '3', 'Arizona', 'c-20107261');
insert into bill values('t-201710', '4.08', '2017', '07', '23', '2',  '2.16', '3', 'Oregon', 'c-20102221');
insert into bill values('t-201711', '4.12', '2018', '02', '14', '1',  '4.73', '1', 'Ohio', 'c-20108132');
insert into bill values('t-201712', '2.76', '2018', '02', '26', '3',  '4.23', '2', 'Seoul', 'c-20101932');
insert into bill values('t-201713', '3.18', '2018', '03', '18', '3',  '6.31', '2', 'Busan', 'c-20108118');
insert into bill values('t-201714', '7.14', '2018', '03', '29', '1',  '1.23', '1', 'Daegu', 'c-20103013');
insert into bill values('t-201715', '1.73', '2018', '04', '23', '2',  '2.32', '1', 'Daezeon', 'c-20106912');
insert into bill values('t-201720', '2.89', '2018', '02', '27', '3',  '4.1', '2', 'Seoul', 'c-20108118');
```

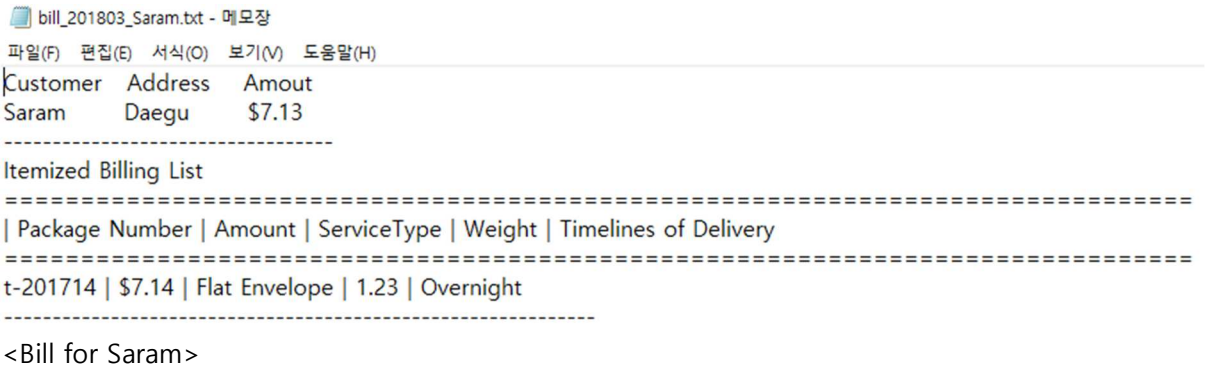
여기서 customer table과 join을 해준뒤 사용자의 이름, 주소, 총 가격을 포함하는 row를 출력해준 뒤에 각각 package별로 package_ID, price, package_type, weight, payment_type을 포함하는 row또한 만들어주었다. 만들어진 Snow의 2017년 2월의 영수증과 Saram의 2018년 3월의 영수증은 다음과 같이 bill_201702_Snow, bill_201803_Saram으로 만들어졌다.


bill_201702_Snow.txt - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
Customer   Address    Amout
Snow       Utah       $4.12
---------------------------------
Itemized Billing List
================================================================================
| Package Number | Amount | ServiceType | Weight | Timelines of Delivery
================================================================================
t-201708 | $1.99 | Flat Envelope | 4.66 | More than two days
t-201721 | $2.13 | Small Box | 5.13 | Second day
------------------------------------------------------------
```

<Bill for Snow>


bill_201803_Saram.txt - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
Customer   Address    Amout
Saram      Daegu      $7.13
---------------------------------
Itemized Billing List
================================================================================
| Package Number | Amount | ServiceType | Weight | Timelines of Delivery
================================================================================
t-201714 | $7.14 | Flat Envelope | 1.23 | Overnight
------------------------------------------------------------
```

<Bill for Saram>

원하던 영수증 정보를 type별로 정확히 출력해주므로 정상적으로 수행되고 있음을 알 수 있다.