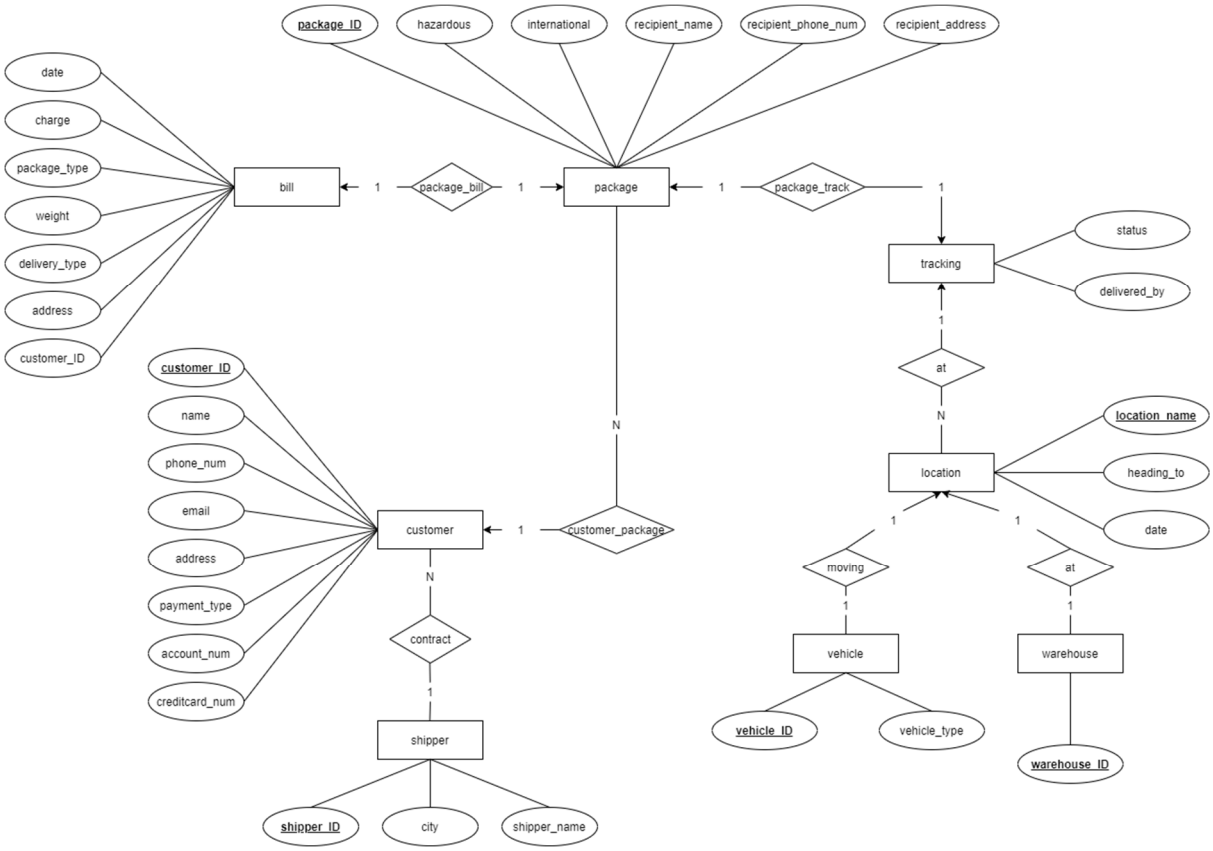
CSE4110 – Database System

project 1

컴퓨터공학과 20161640

정동혁

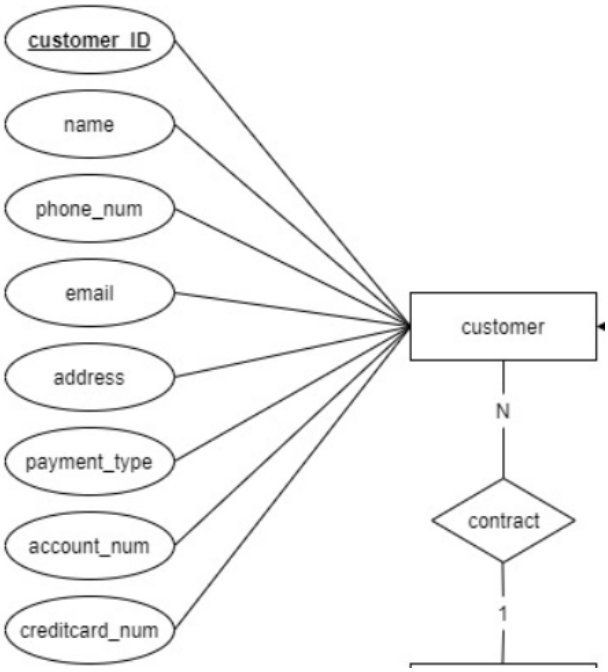
1. ER Diagram



1. **Entity set:**
   - **1-1). customer**
   - **1-2). shipper**
   - **1-3). package**
   - **1-4). bill**
   - **1-5). tracking**
   - **1-6). location**
   - **1-7). vehicle**
   - **1-8). warehouse**

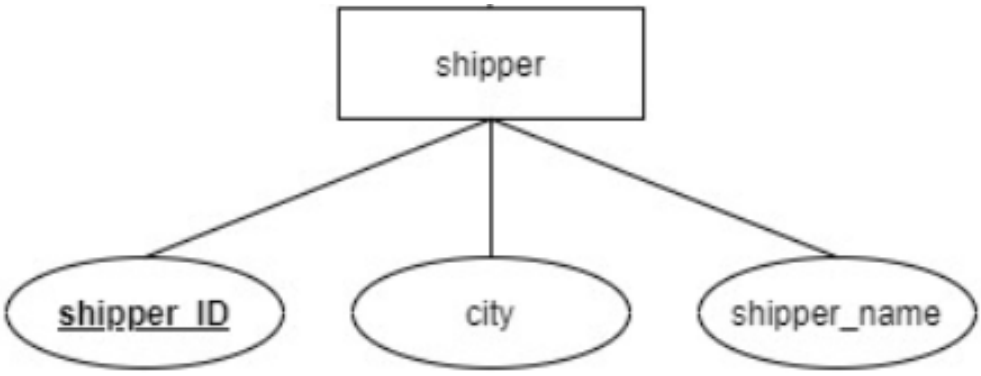
2. Attribute

2-1). customer : customer_ID(PK), shipper_ID(FK), name, phone_num, email, address, payment_type, account_num, creditcard_num



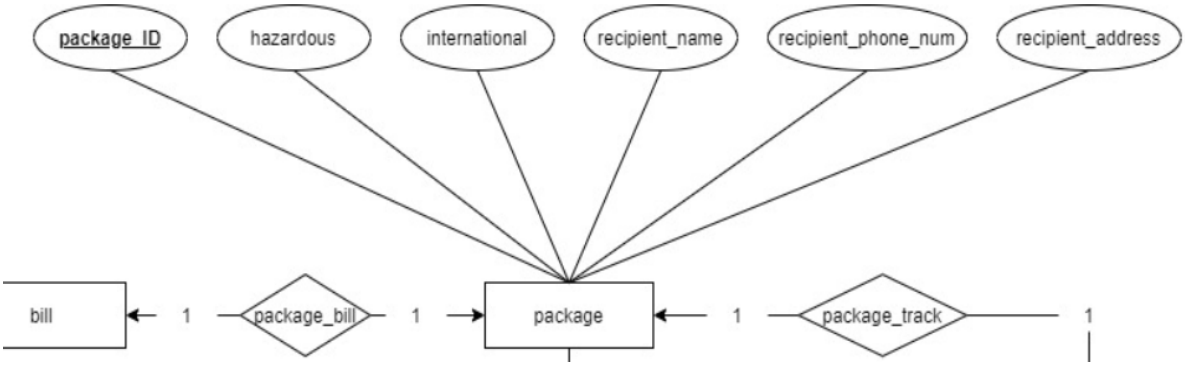
소비자의 경우 ID라는 각 사람마다 고유번호를 만들고 PK로 지정하였다. 또한 통상적으로 배송회사의 소비자가 가져야할 정보인 소비자 이름, 연락처, 주소, 결제방식과 결제수단의 정보를 attribute로 생성하였다.

2-2). shipper: shipper_ID(PK), city, shipper_name



배송원의 경우 ID를 통해 각 배송원을 구분할 수 있게 하였고, 그 배송원이 근무하는 근무지역, 배송원의 이름등을 attribute로 생성하였다.

2-3). package: package_ID(PK), customer_ID(FK), hazardous, international, recipient_name, recipient_phone_num, recipient_address

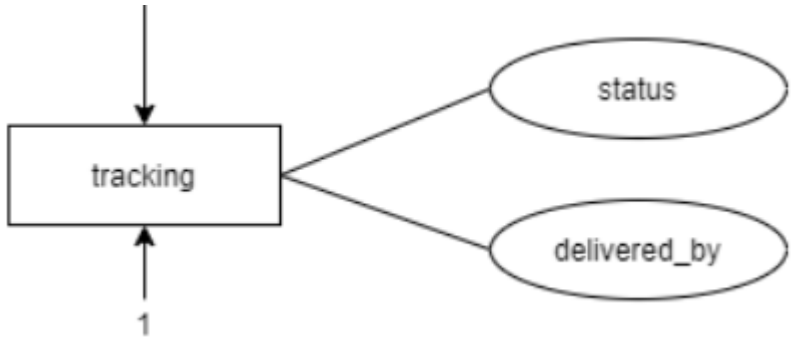


package의 경우 package_ID(운송장번호)를 통해 구분할 수 있게 하였다. hazardous와 international또한 package의 attribute로서 사용하였는데 이유는 이후 domain설정파트에서 언급할 것이다. 추가적으로 일반적으로 택배 배송시 중요한 수령인 이름, 수령인 연락처, 수령인 주소등을 attribute로 기용하였다.

2-4). bill: <u>package_ID(FK)</u>, <u>customer_ID(FK)</u>, date, charge, package_type, weight, delivery_type, address, customer_ID

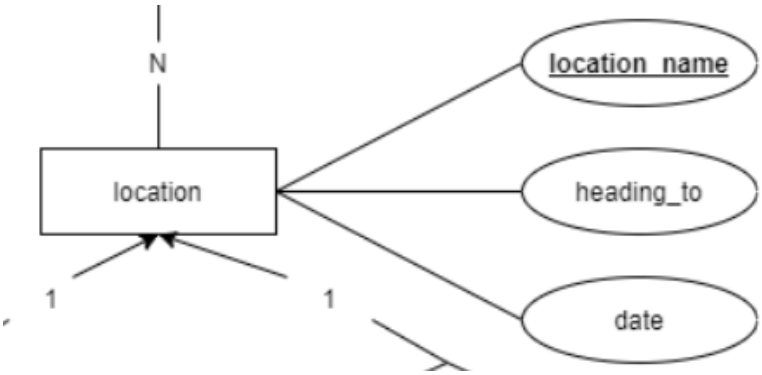
bill의 경우 package로부터 받아온 package_ID를 통해 구분할 수 있게 하였다. 또한 배송비 결제시간, 결제금액, 이용한 서비스타입(package_type, weight, delivery_type)과, 주소, 소비자ID등을 attribute로 기용하였다. customer에서 기용한 attribute인 address를 다시 기용한 이유는 Oracle에서 사용자에게 bill을 제공할 때 위와 같은 bill table을 그대로 bill로서 받게 되어도 요구되는 모든 정보가 포함되어야 하기 위함이다.

2-5). tracking: package_ID(FK), customer_ID(FK), status, delivered_by



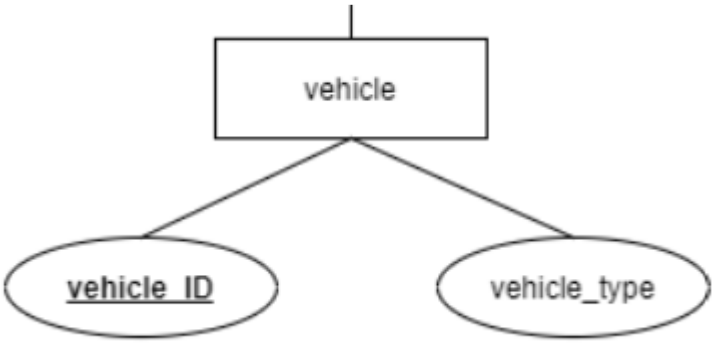
tracking의 경우 package로부터 받아온 package_ID를 통해 구분할 수 있게 하였다. 또한 통상적으로 배송추적을 할때의 배송의 상태(normal, delayed)와 배달담당자의 이름을 표시하기 위해 status와 delivered_by를 attribute로 기용하였다. 일반적인 tracking이란 결국 배송추적을 가능하게 하는 entity여야 하지만 위치정보는 location에서 insert되는 구조로 er-model을 설계하였다.

2-6). location: location_address(PK), package_ID(FK), customer_ID(FK), date



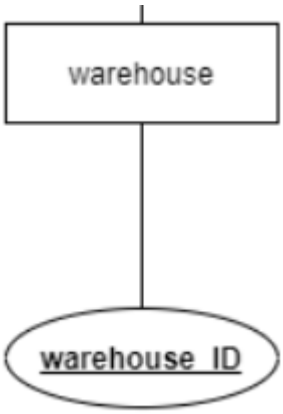
location의 경우 배송품의 현재 위치(location_name)와 tracking서비스로부터 받아온 package_ID를 통해 구분할 수 있게 하였다. 또한 date를 attribute로 기용함으로서 어떠한 시간에 어디에 package가 존재하는지 알 수 있게 하였다. 마지막으로 heading_to를 attribute로서 사용한 이유는 date와 location_name을 통해서 과거와 현재까지의 배송물위치정보는 입력이 가능하지만 현재 목적지는 표현할 수 없으므로 따로 attribute를 생성하였다.

2-7). vehicle: vehicle_ID(PK), package_ID(FK), customer_ID(FK), location_name(FK), vehicle_type



vehicle의 경우 현재 운행중인 vehicle의 고유번호인 vehicle_ID와 location으로부터 받아온 package_ID, location_name을 통해서 구분할 수 있게 하였다. 추가적으로 vehicle_type은 기업에서 요구한 현재 운송하는 기계의 종류가 truck인지 plane인지 알기위해서 attribute로서 기용하였다.
(warehouse는 따로 구현하였다.)

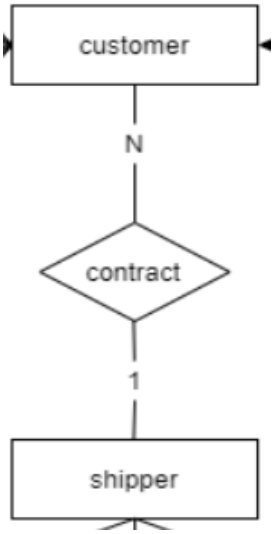
2-7). warehouse: warehouse_ID(PK), package_ID(FK), customer_ID(FK), location_name(FK), vehicle_type



warehouse의 경우 현재 warehouse의 고유번호인 vehicle_ID와 location으로부터 받아온 package_ID, location_name을 통해서 구분할 수 있게 하였다. warehouse를 vehicle과 다른 entity로 만든 이유는 이후 relation파트에서 언급할 것이다.

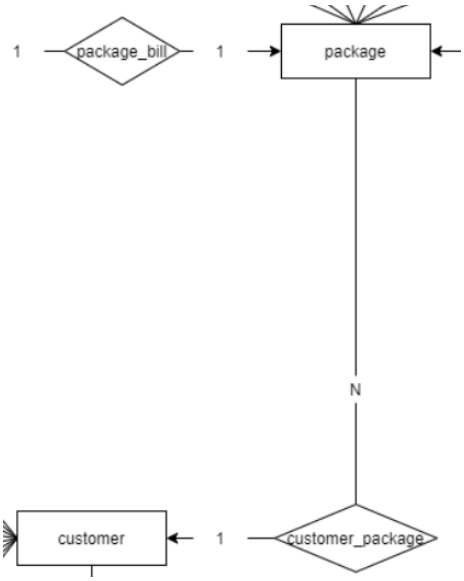
3. Relation & Mapping

3-1). customer, shipper



명세서에서 customer는 shipper와의 개인 contract가 있을 수도 있다 하였으므로 둘간의 relation을 contract라고 명명하였다. 또한 한 도시(동)당 담당하는 shipper가 존재하고 그러한 shipper는 여러명의 customer의 배송을 담당하므로 MANY TO ONE relationship으로 설계하였다.

3-2). customer, package

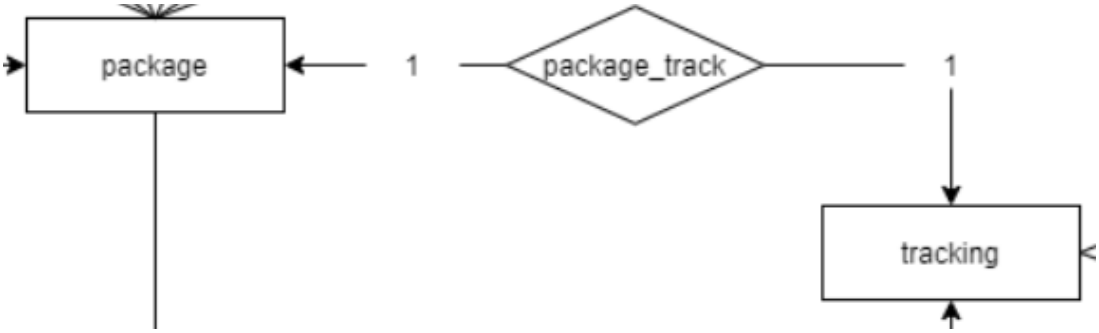


일반적으로 customer한명당 여러개의 package배송을 시킬 수 있지만 한 개의 package배송을 여러명의 customer가 시킬 수 없으므로 MANY TO ONE relationship으로 설계하였다.

3-3). package, bill

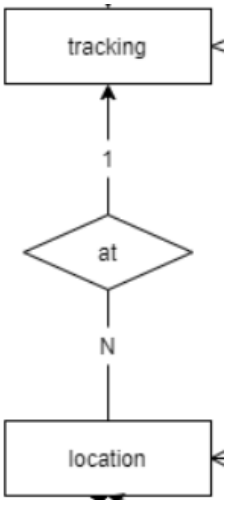
명세서의 query부분에서 언급된 bill의 조건을 만족하기위해 package 한 개당 한 개의 bill이 만들어지게 설계하기 위해 ONE TO ONE relationship으로 설계하였고 package_bill로 명명하였다.

3-4). package, tracking



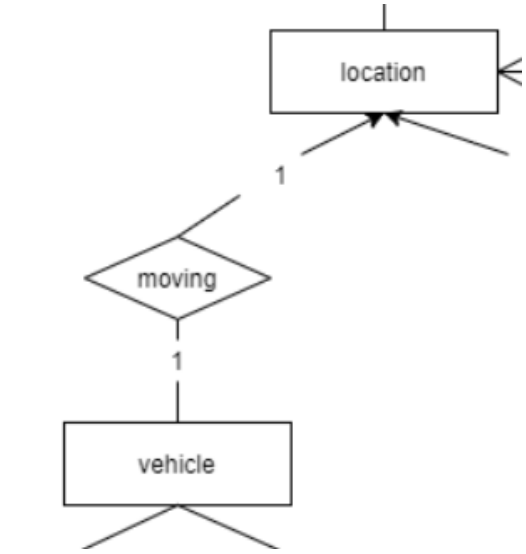
하나의 package는 하나의 tracking정보만을 담을 수 있고, 하나의 tracking 정보는 오로지 하나의 package의 정보가 될 수 있으므로 ONE TO ONE relationship으로 설계하고 package_track으로 명명하였다.

3-5). tracking, location



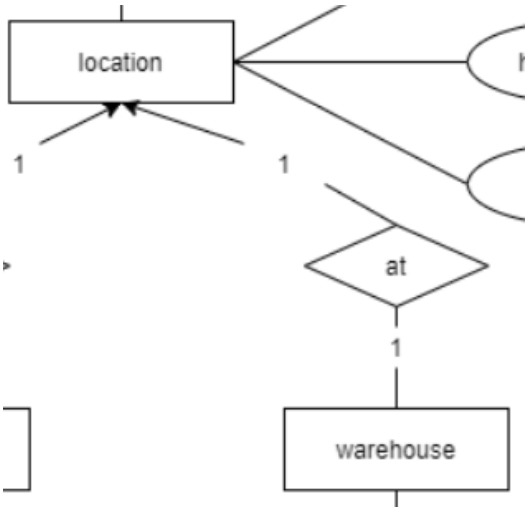
일반적인 배송추적의 경우 과거의 정보와 현재의 정보를 표시하기 때문에 하나의 tracking은 여러 개의 location정보를 가질 수 있다. 그러므로 MANY TO ONE relationship으로 설계하고 ~위치에 존재함을 의미하므로 at으로 명명하였다.

3-6). location, vehicle



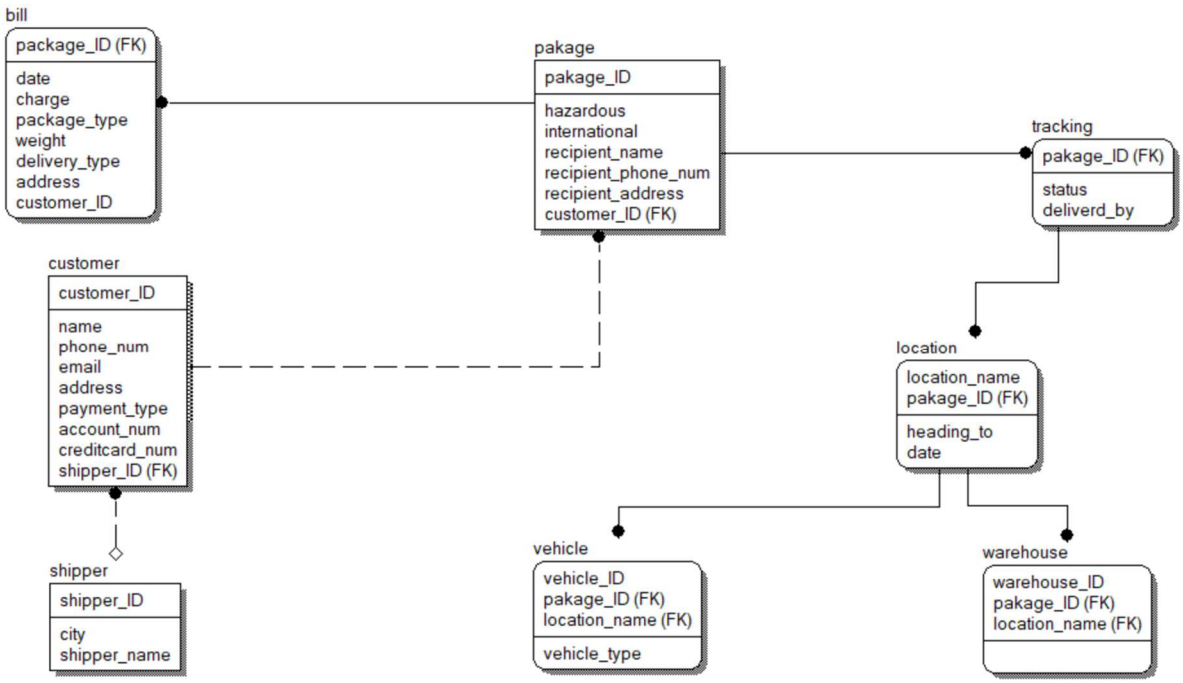
명세서에 명시된 바에 의하면 배송정보를 vehicle의 standpoint로도 열람이 가능해야 한다 하였으므로 location과 vehicle을 moving relation으로 연결하였다. 보통 배송의 경우 한 위치에서 운송기계는 하나로 정해져 있으므로 ONE TO ONE relationship으로 설계하였다.

3-7). location, warehouse

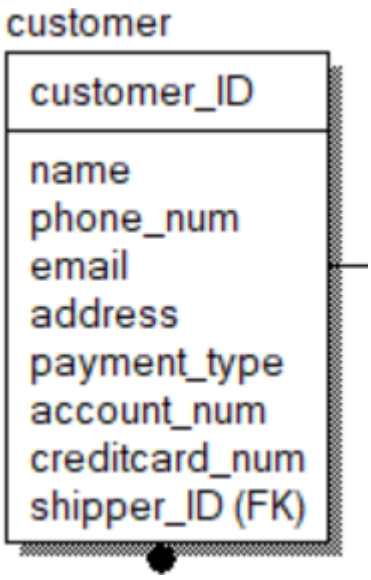


명세서에서 명시된 바로는 결국 운송품을 운반하는 위치는 truck, plane, warehouse뿐이지만 warehouse경우는 배송품을 임시저장하는 용도로서 사용되고 , vehicle의 경우에는 배송품을 이동하기 위한 이동수단이므로 relation의 차이가 존재한다 생각해서 다른 relation(at)으로 연결해주었다. 위의 vehicle과 마찬가지로 이 relationship또한 ONE TO ONE relationship 이다.

4. Logical model

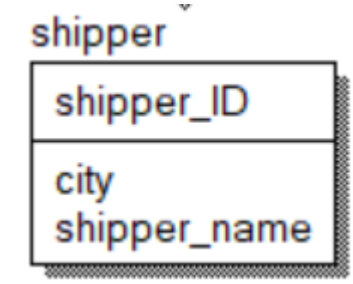


4-1). customer



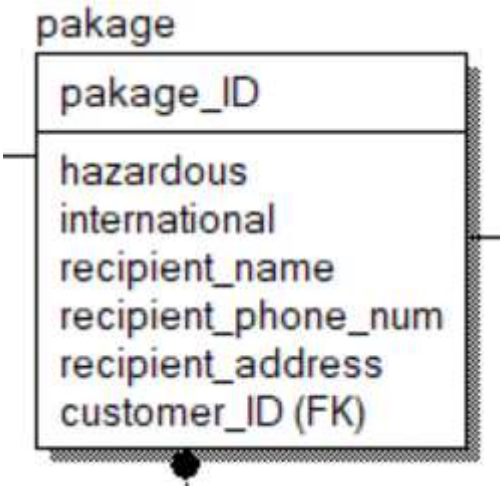
소비자, 고객을 의미하는 entity이다. 위의 er-model에서 언급한 attribute들을 기용하였으며 고객이 shipper와 contract가 되어있는 shipper가 존재할 수 있으므로 shipper_ID를 FK로 받아들인다. 회원은 ID만으로도 구분될 수 있기 때문에 customer_ID만을 PK로 결정하였다. 나머지 속성들은 모두 배송주문을 위해 필요한 정보들을 담았다.

4-2). shipper



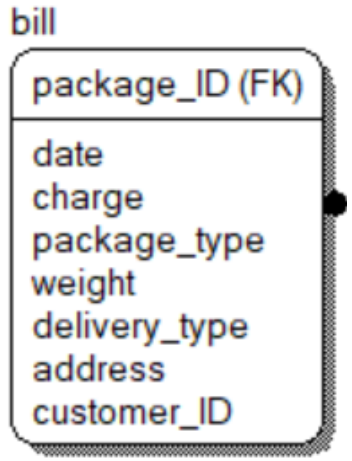
배송담당자를 의미하는 entity이다. 명세서에 어떠한 고객들은 배송부와 계약을 했을수도있다 라는 언급이 있었기에 entity로 기용하였다. 배송부 또한 ID만으로도 구분될 수 있기 때문에 shipper_ID만을 PK로 결정하였다. 다른 속성으로는 근무도시, 배송부 이름등이 있다.

4-3). package



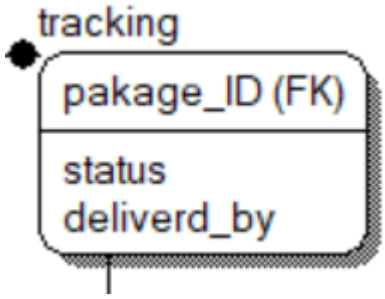
운송하는 물건을 의미하는 entity이다. 일반적으로 배송은 운송장번호로도 구분이 되어지기 때문에 pacakge_ID를 PK로 결정하였고 customer의 ID를 non-identifying relation을 통해 FK로 받아오게 하였다. 또한 기업의 물건 관리를 위해서 운송하는 물건의 정보로서 hazardous와 international을 속성으로 담아주었다.

4-4). bill



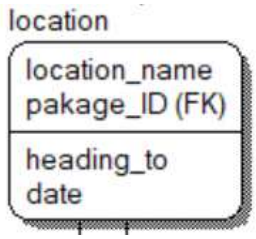
영수증을 의미하는 entity이다. 한 배송에 대한 영수증의 경우 운송장번호로 구분될 수 있으므로 package_ID를 identifying relation을 통해 FK로 받아왔다. 명세서에 언급된 바로는 SQL담당회사인 oracle에서 bill table을 제공하면 바로 일반적인 사람들이 알아볼 수 있고 주소, 가격, 서비스타입, 주소, 소비자ID 등을 알 수 있어야한다 하였으므로 data redundancy가 존재하여도 bill table의 속성으로 담아주었다.

4-5). tracking



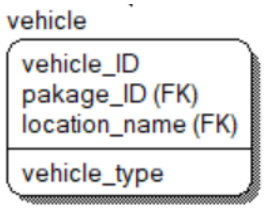
배송추적을 의미하는 entity이다. 한 배송에 대한 추적을 할 때 운송장번호로 구분될 수 있으므로 package_ID를 identifying relation을 통해 FK로 받아왔다. 추가적인 속성으로는 배송이 정상적인 상태로 되어지고 있는지, 안되고있는지(ex)truck crash)를 확인 할 수 있는 status와 배송담담자 등이 있다.

4-6). location



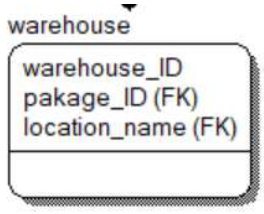
배송추적중 장소(위치)를 의미하는 entity이다. 한 배송에 대한 위치는 location과 운송장번호로 구분될 수 있으므로 location_name을 PK로 사용하고 package_ID를 identifying relation을 통해 FK로 받아왔다. 추가적인 속성으로는 앞에서 언급하였듯이 현재 향하고 있는 정보를 담기 위한 heading_to와 특정위치에서의 날짜,시간등을 포함한 정보인 date 등이 있다.

4-7). vehicle



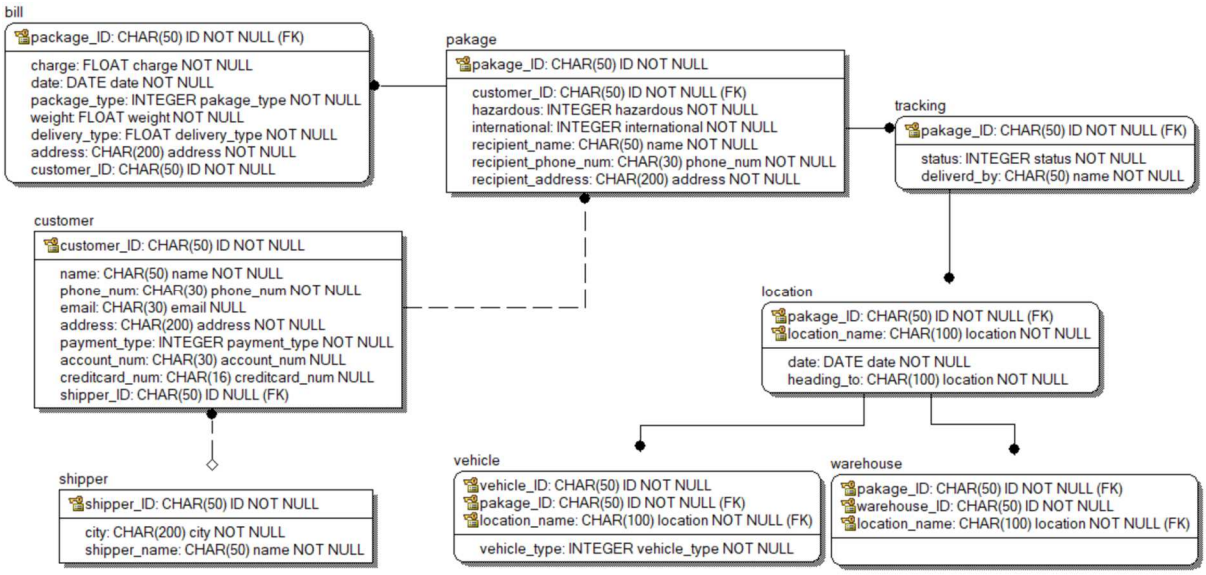
운송기계를 의미하는 entity이다. 각각의 운송기계를 구분하기 위해 운송기계의 고유번호인 vehicle_ID를 PK로서 기용하고 그 특정 기계가 위치한 장소, 운송장정보를 알기위해 package_ID와 location_name을 identifying relation을 통해 FK로 받아왔다. 추가적인 속성으로는 기계의 종류를 알려주는 vehicle_type등이 있다.

4-8). warehouse



창고를 의미하는 entity이다. 위의 vehicle과 비슷하게 구현하였다. 창고를 구분하기 위해 warehouse_ID를 PK로서 기용하고 특정 창고가 위치한 장소, 운송장정보를 알기위해 package_ID와 location_name을 identifying relation을 통해 FK로 받아왔다

5. 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.

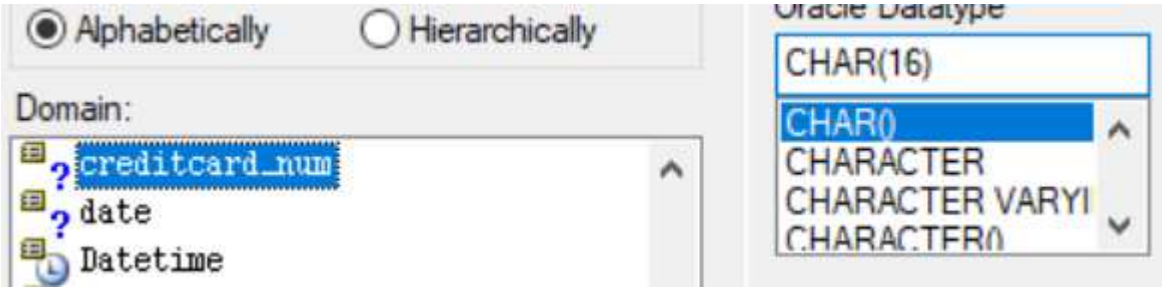
5-3). charge

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

5-4). city

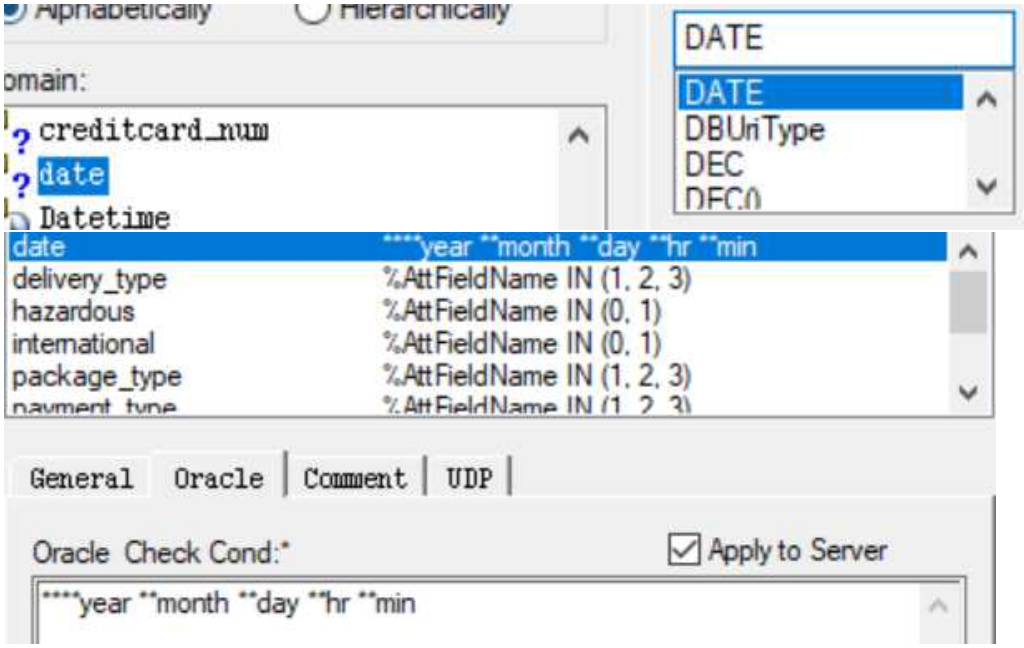
도시를 의미하는 city의 경우 구글에서 찾아본 결과 가장 긴 도시이름의 길이가 200을 넘지 않았으므로 CHAR(200)으로 지정해주었다. 일하는 곳의 도시이름이 없을수 없으므로 NOT NULL로 해주었다.

5-5). creditcard_num



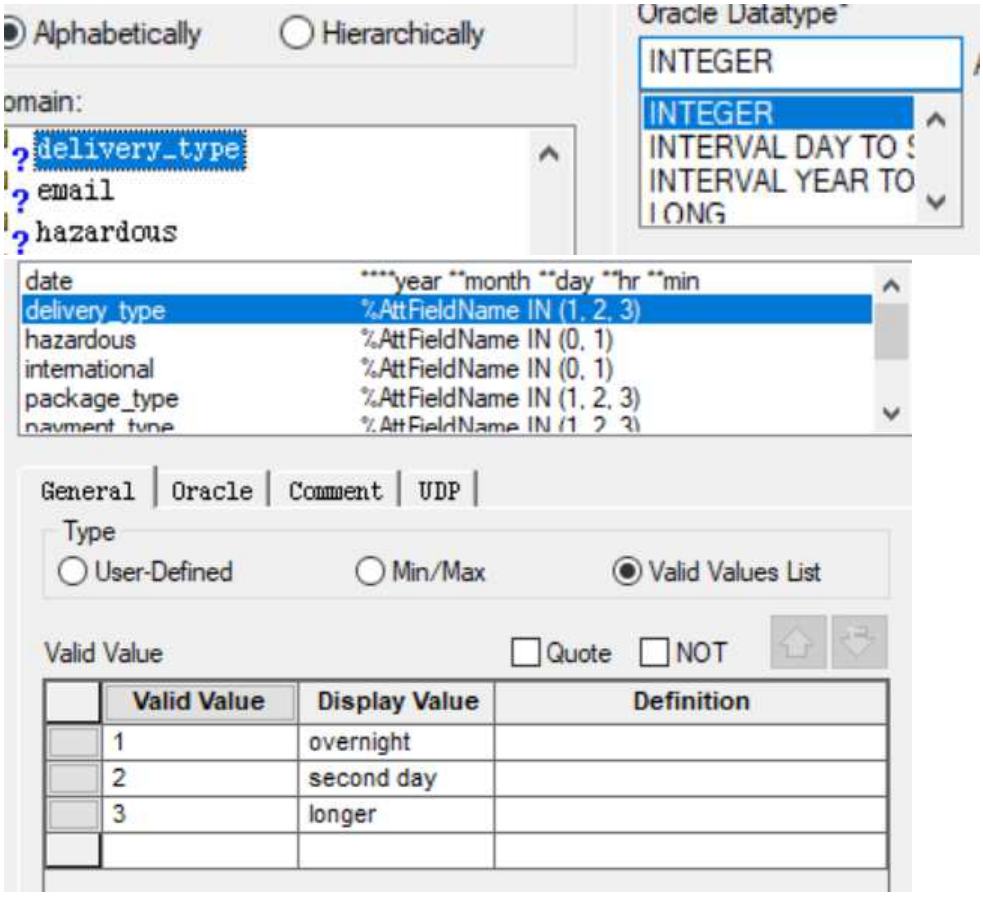
카드번호를 의미하는 creditcard_num의 경우 국제적으로 사용되는 Visa, MasterCard와 같은 카드들은 모두 16자리의 숫자이므로 CHAR(16)으로 지정해주었다. 카드결제방식이 아닌 다른 결제방식으로 사용할 수 있으므로 NULL값을 허용해 주었다.

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

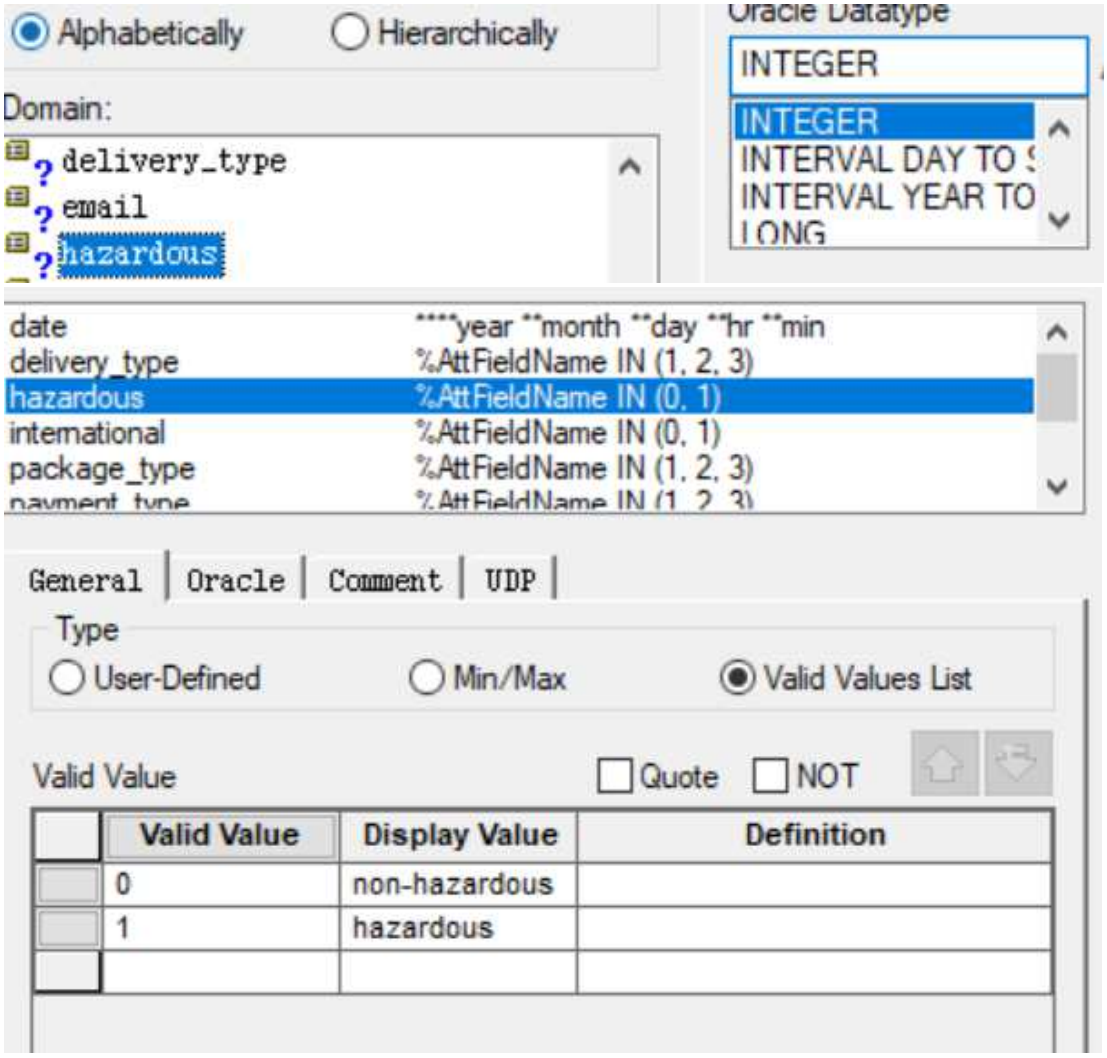


배송예정시간을 의미하는 delivery_time의 경우에는 INTERGER typed으로 선언해주었는데 validation rule을 적용하여 1의 값을 가질 경우 overnight, 2의 값을 가질 경우 second day, 3의 값을 가질 경우 longer를 의미하게 해주었다. 3가지 type중 무조건 한 가지를 만족해야하므로 NOT NULL로 해주었다.

5-8). email

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

5-9). hazardous

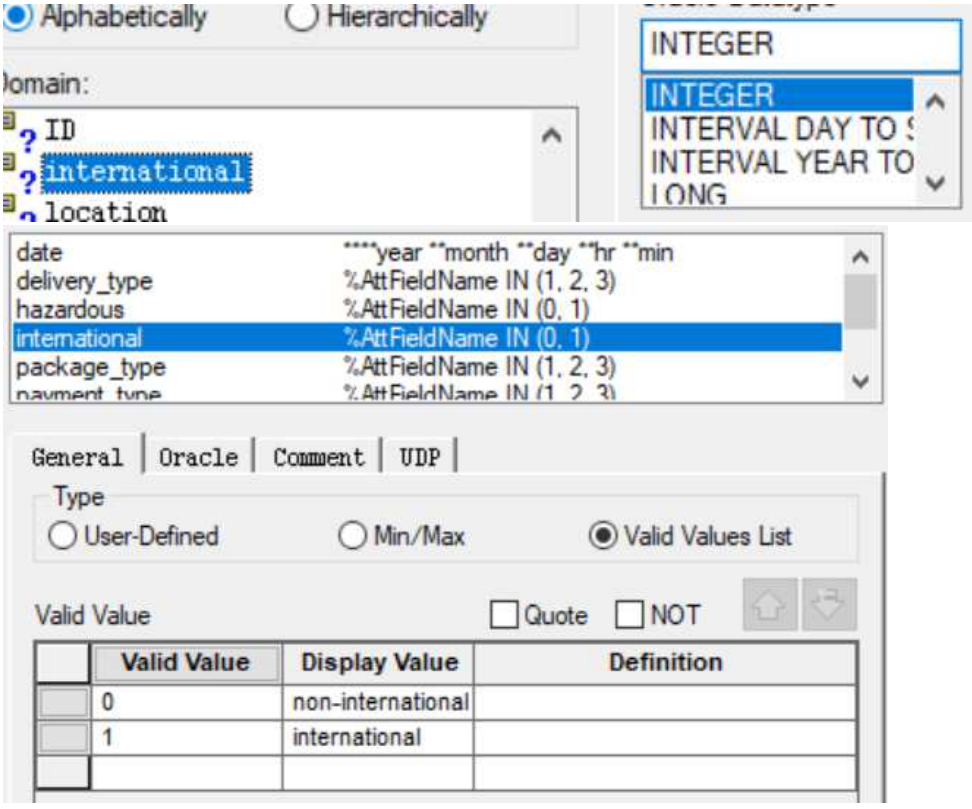


위험물을 뜻하는 hazardous의 경우 0또는 1의 값을 가지는 INTERGER type으로 선언해주었다. 0은 위험물이 아님을 뜻하는 non-hazardous, 1은 위험물을 뜻하는 hazardous를 의미하게 해주었다. package의 hazardous여부는 non-hazardous혹은 hazardous둘 중 하나의 값을 무조건 가지게 되므로 NOT NULL로 해주었다.

5-10). ID

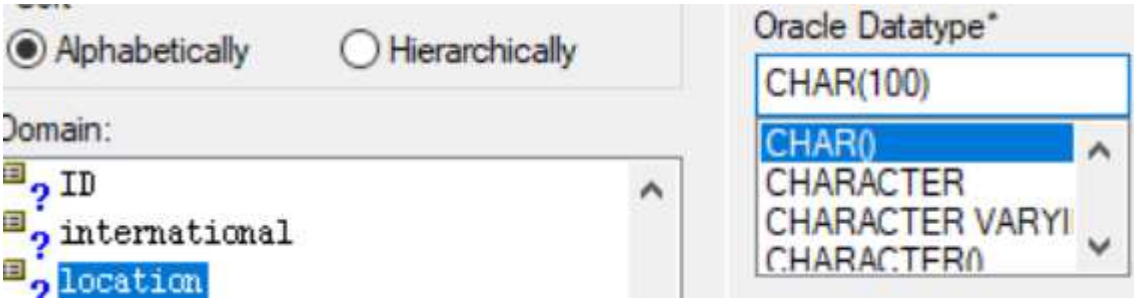
소비자, 배송원, 운송수단 등을 name을 가지고는 unique하게 구별해 낼 수 없으므로 ID값을 만들어주었고, 최대 50길이의 CHAR type으로 선언해주었다. ID는 구별요소이기 때문에 무조건적으로 가져야하므로 NOT NULL로 해주었다 NOT NULL의 예외로는 customer entity에 존재하는 shipper_ID의 경우 배송부와 손님간의 계약이 존재하지 않을 수 있기 때문에 NULL값을 허용해주었다.

5-11). international



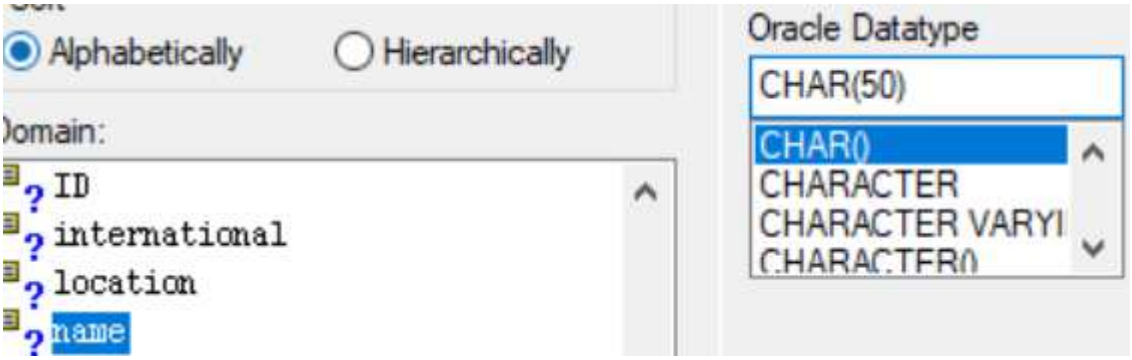
해외배송물을 뜻하는 international의 경우 0또는 1의 값을 가지는 INTERGER type으로 선언해주었다. 0은 국내배송을 뜻하는 non-international, 1은 해외배송물을 뜻하는 international을 의미하게 해주었다. package의 international여부는 non-international혹은 international 둘 중 하나의 값을 무조건 가지게 되므로 NOT NULL로 해주었다.

5-12). location



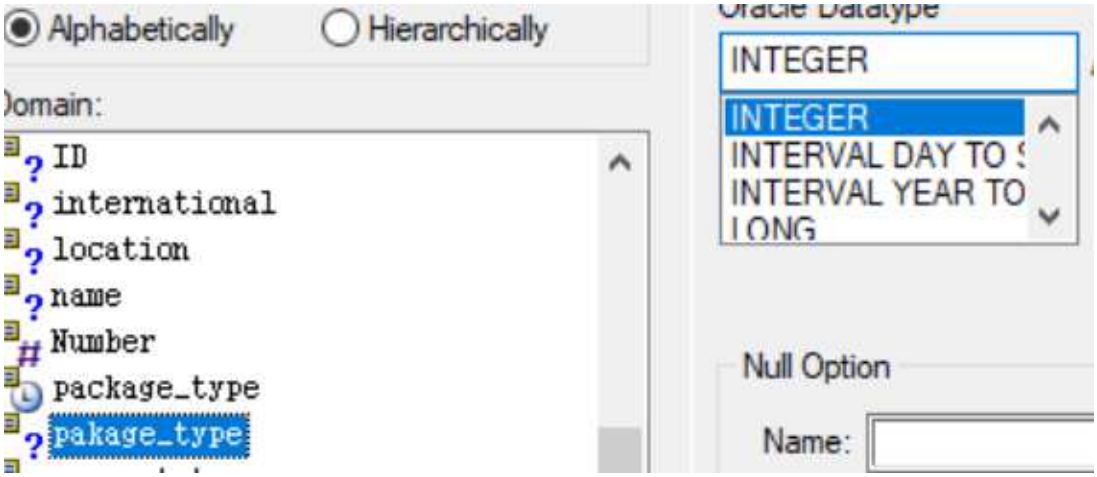
현재의 장소, 위치 등을 표현하는 location의 경우는 CHAR(100)을 통해 길이를 100으로 제한해주었다. 또한 위치정보는 무조건적으로 가지고 있어야 하는 값이므로 NOT NULL로 해주었다.

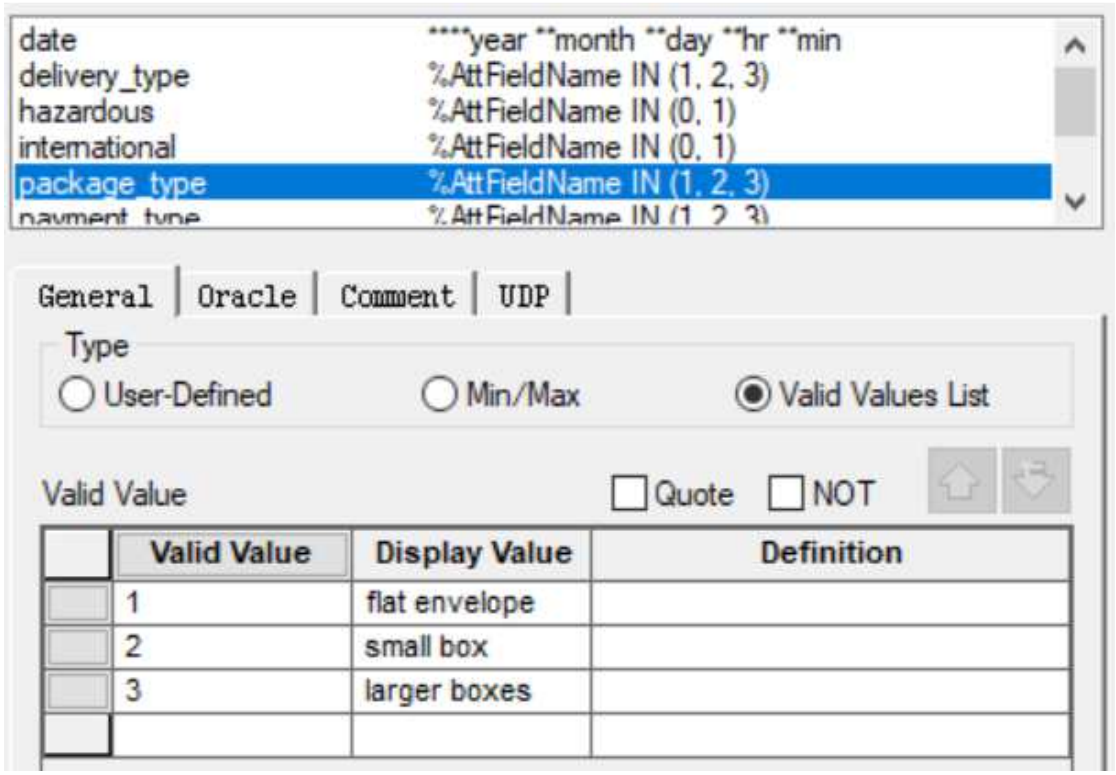
5-13). name



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.

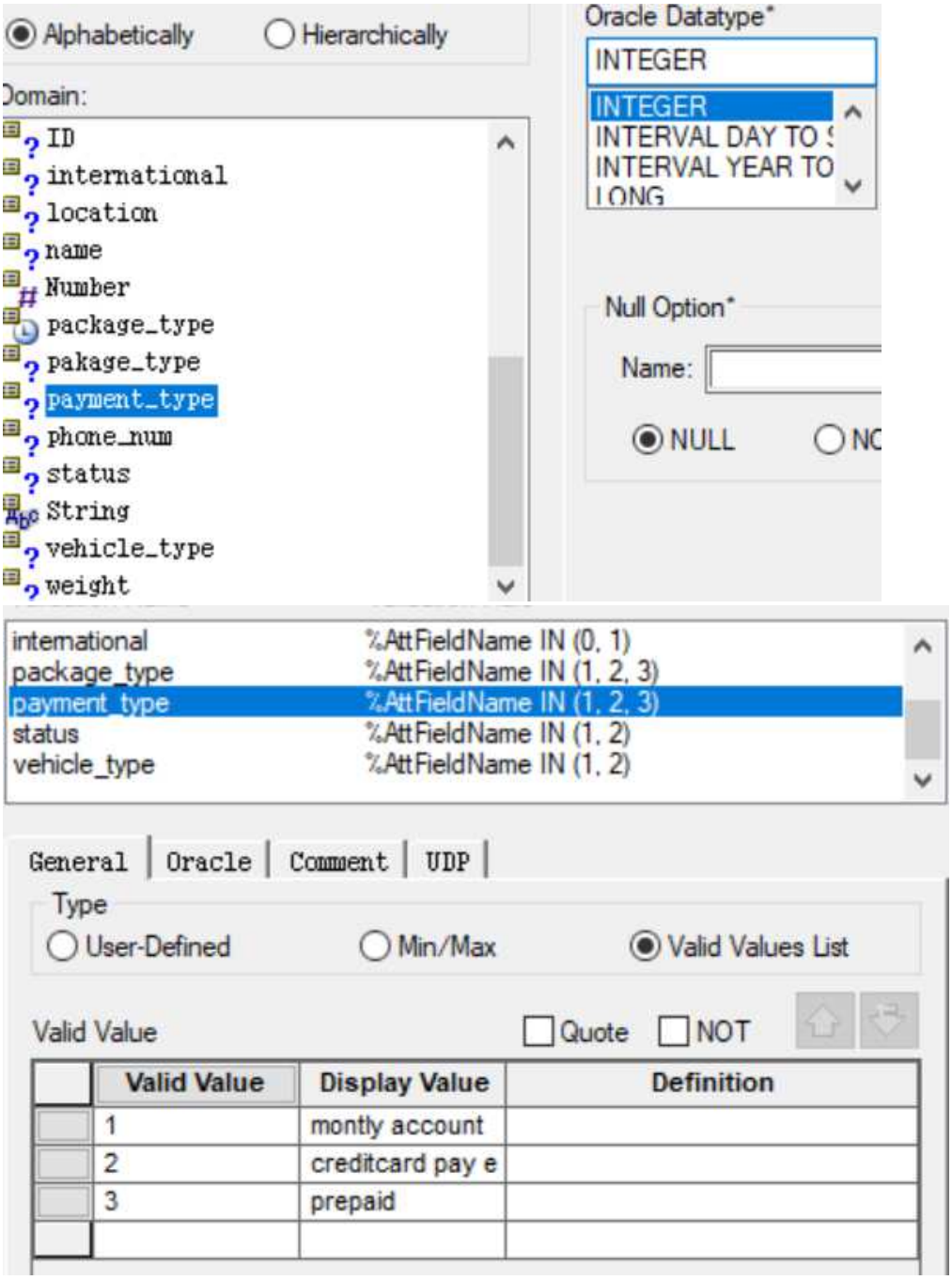
5-14). package_type





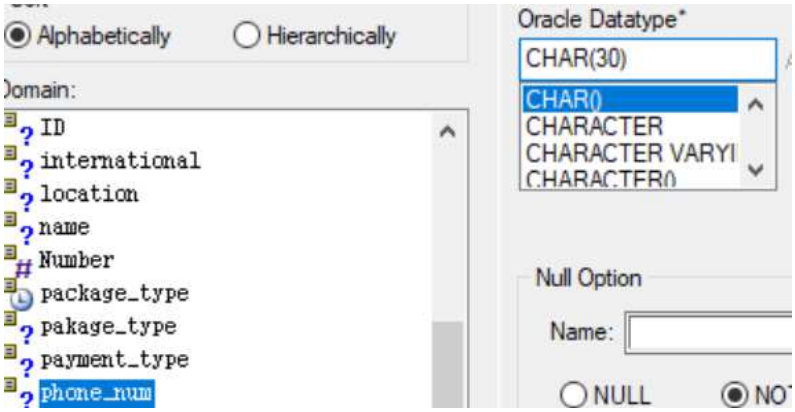
package의 포장형태를 의미하는 package_type의 경우는 validation rule을 이용하여 1, 2, 3의 값을 가질 수 있는 INTERGET type으로 선언해주었다. 각각 값을 살펴보면, 1의 값을 가지게 되면 flat envelope, 2의 값을 가지게 되면 small box, 3의 값을 가지게 되면 larger boxes를 의미하게 하였다. 배송품의 포장형태는 무조건적으로 위의 값들중 하나의 값을 가져야 되기 때문에 NOT NULL로 해주었다.

5-15). payment_type



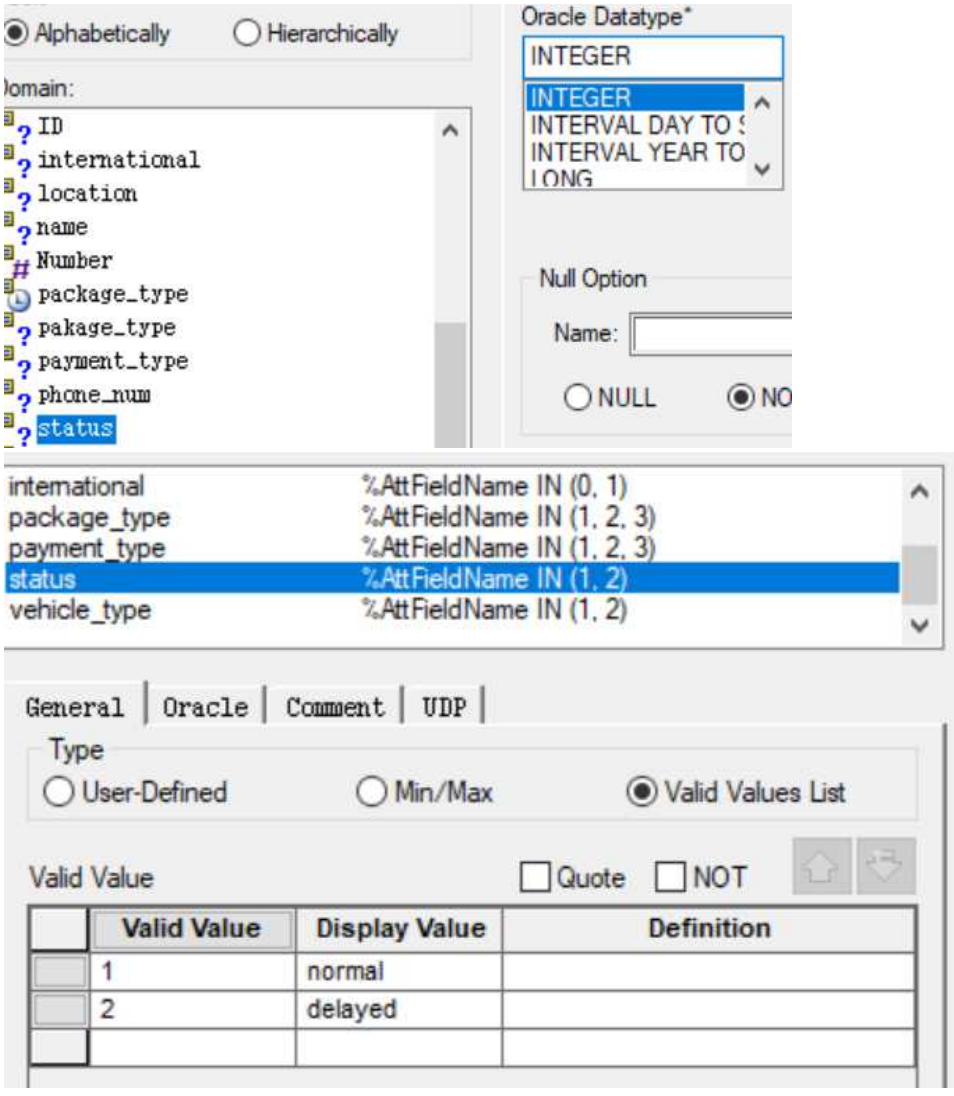
결제방식을 의마하는 payment_type의 경우는 validation rule을 이용하여 1, 2, 3의 값을 가질 수 있는 INTEGER type으로 선언해주었다. 각각 값을 살펴보면, 1의 값을 가지게 되면 montly_account(달마다 account로 pay), 2의 값을 가지게 되면 creditcard_pay_each(배송시킬때마다 카드결제), 3의 값을 가지게 되면 prepaid(선결제)를 의미하게 하였다. 배송의 결제방식은 무조건적으로 위의 값들중 하나의 값을 가져야 되기 때문에 NOT NULL로 해주었다.

5-16). phone_num



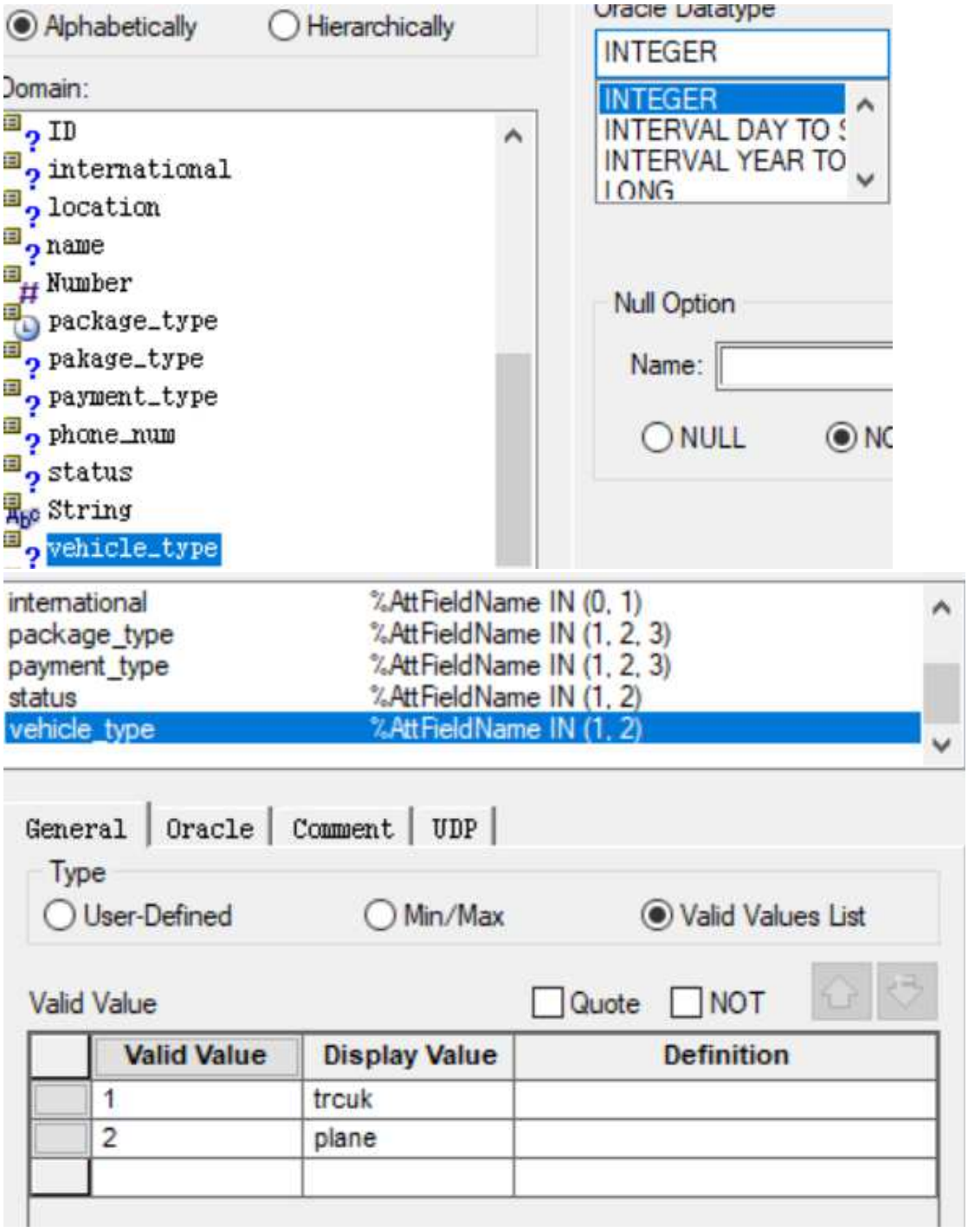
전화번호를 의미하는 phone_num의 경우 전 세계적으로 전화번호가 30자를 넘지 않기 때문에 30자로 길이를 제한하였고 배송 진행시에 수령인과 발송인 모두의 전화번호는 필수적이므로 NOT NULL로 해주었다.

5-17). status



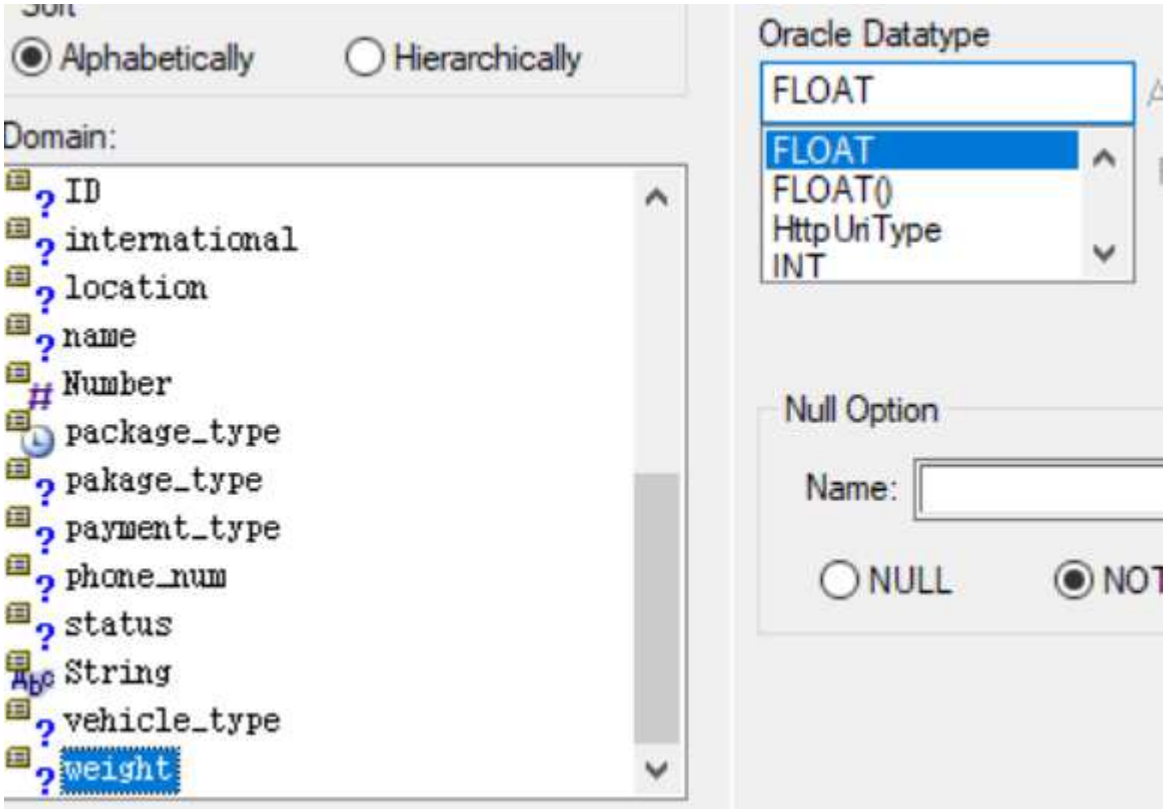
현재 배송상태를 의미하는 status의 경우는 validation rule을 이용하여 1, 2의 값을 가질 수 있는 INTERGER type으로 선언해주었다. 각각 값을 살펴보면, 1의 값을 가지게 되면 normal(정상배송상태), 2의 값을 가지게 되면 delayed(배송지연상태[ex)truck crash])를 의미하게 하였다. 배송은 항상 정상배송 아니면 지연배송이므로 NOT NULL로 해주었다.

5-18). vehicle_type



현재 운송중인 기계의 종류를 의미하는 vehicle_type의 경우는 validation rule을 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.

6. **specification condition**

- There are different kinds of service possibly based upon the type of package(flat envelope, small box, larger boxes), the weight of the package, and the timeliness of delivery (overnight, second day, or longer).

- Some customers have a contract with the shipper and bill their shipments to an account number. They are billed monthly. Other customers are infrequent customers and pay with a credit card. Certain shipments are prepaid, as is might be the case of someone is returning something that was purchased by phone or Internet (e.g. returning clothes that don't fit, or returning malfunctioning electronics, returning damaged book).

- For the most part, the shipping company does not care what is being shipped. However, there are cases where it matters. Some examples include
  - hazardous materials
  - international shipments, for which a customs declaration stating the contents and their value is needed

- The company needs to track packages from the time the customer drops it off (or it is picked up by the company) until the time is is delivered and signed for. Take a look at the online tracking offered by various shipping companies to get an idea of how this service works. If you are having something shipped to you, you'll find you can get every little detail of where the package is, where it has been, and to where it is currently headed. Beyond what the customer sees, the company itself needs to know on which truck or plane or warehouse the package is at any point in time.

- Tracking is not just an "in the present" issue. The company may want to look back in time and find out where the package was yesterday, for example. It may also want to look at data from the standpoint of a truck or warehouse.

- There are other aspects to the operation of the company besides package tracking such as the routing of trucks and planes, the assignment of staff to them, etc. For this project, we'll consider only the package handling and billing aspects of the database.

1). bill에 package_type, weight, delivery_type을 attribute로 기용함으로서 service type을 알 수 있게 하여 첫 번째 조건을 만족시켰다.

2). shipper라는 entity를 생성하여 customer와 contract relation을 만들어주고 customer에 payment_type, account_num. creditcard_num을 추가함으로서 두 번째 조건을 만족시켰다.

3). package에 위험성여부(hazardous)와 국제배송여부 (international)을 attribute로 기용함으로서 세 번째 조건을 만족시켰다.

4). package entity에 tracking, location, vehicle, warehouse entity를 연결해줌으로서 배송품의 위치와 시간, 운송수단, 배송상태 등을 알 수 있게 하였으므로 네 번째 조건을 만족시켰다.

5). location entity가 시간과 장소의 정보를 attribute로서 기용하여 원하는 시점(yesterday, now등)의 배송이 추적가능하게 설계함으로서 다섯 번째 조건을 만족시켰다.

6). DB모델을 설계할 때 운송품의 배송과 결제에 중점을 두고 설계하였으므로 여섯 번째 조건을 만족시켰다.

7. query

- Assume truck 1721 is destroyed in a crash. Find all customers who had a package on the truck at the time of the crash. Find all recipients who had a package on that truck at the time of the crash. Find the last successful delivery by that truck prior to the crash.
- Find the customer who has shipped the most packages in the past year.
- Find the customer who has spent the most money on shipping in the past year.
- Find those packages that were not delivered within the promised time.
- Generate the bill for each customer for the past month. Consider creating several types of bills.
  - A simple bill: customer, address, and amount owed.
  - A bill listing charges by type of service.
  - An itemize billing listing each individual shipment and the charges for it.

Customers like their bills to be readable. While the client will accept a relational table coming from Oracle as the bill, it would be "nice" to have a good-looking bill.

1). **select** customer.name, recipient_name, location_name
from customer, package, vehicle
where vehicle_ID = "TR1721" and package.package_ID = vehicle.package_ID and customer.customer_ID = package.customer_ID

2). **select** T.customer_ID
from bill as T, bill as S
group by customer_ID
having count(T.package_ID)>count(S.package_ID) and T.date like "2019%" and S.date like "2019%"

3). **select** T.customer_ID
from bill as T, bill as S
group by customer_ID
having sum(T.charge)>sum(S.charge) and T.date like "2019%" and S.date like "2019%"

4). **select** pacakge_ID

from tracking

where status = 2

5). simply pop out bill table which bill.data = “2020 03%”.