CSE4110 - Database System

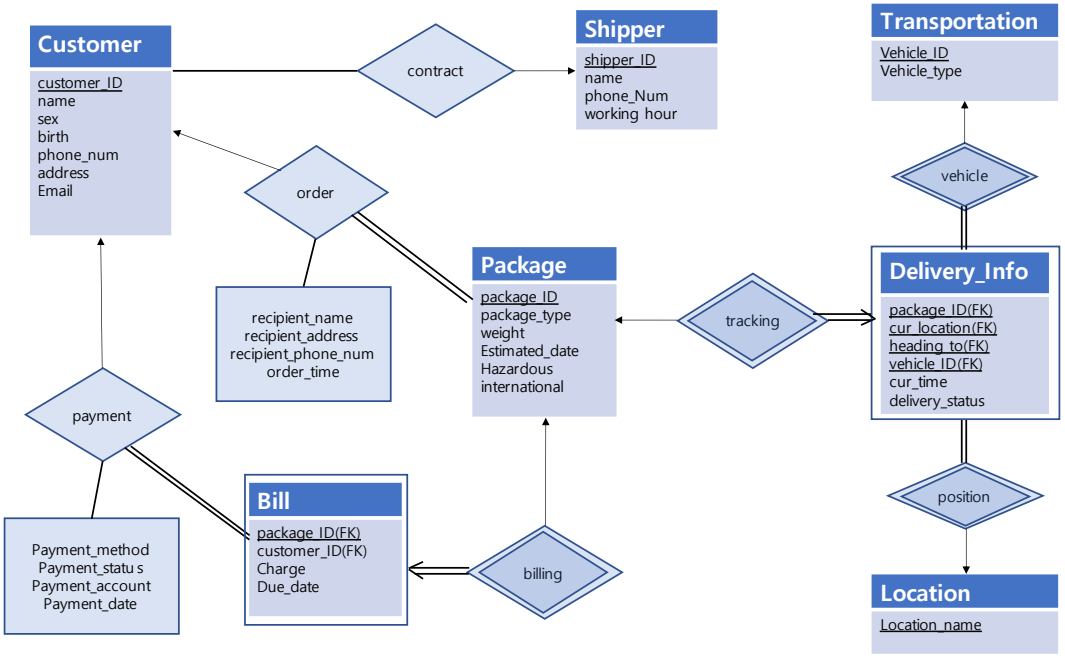
Project1. E-R design and Relational Schema design

20180032

남기동

목차

1.1 E-R Diagram



1.1 Entity Set

1) Customer

Customer_ID: 배달을 요청하는 고객의 unique ID number이며 이를 통해 고객들을 식별한다.

Name: 고객의 이름 정보를 저장한다.

Sex: 고객의 성별을 저장한다

Birth: 고객의 생년월일을 저장한다

phone_num: 고객의 연락처를 저장한다

address: 고객의 실 거주지를 저장한다

email: 고객의 이메일 주소를 저장한다.

2) Shipper

Shipper_ID: 택배기사의 정보를 담고 있으며 primary key 역할을 한다

Name: 택배기사의 이름을 담고 있다.

phone_num: 택배기사의 전화번호를 담고 있다

work_hour: 택배기사가 일하는 시간대를 의미하며 day 근무와 night 근무가 있을 수 있기 때문에 설정한 속성이다.

3) Package

이번 DB에서 핵심이 되는 entity로 패키지의 정보를 저장하기 위해 설정된 entity이다.

Package_ID: 각 패키지마다 가지고 있는 고유 일련 번호로 패키지를 식별하게 하는 속성이기 때문에 primary key로 설정된다.

Package_type: 해당 패키지가 어떤 형태, 종류인지를 나타내는 속성으로 다음과 같은 값들로 명세서에서 언급된 것과 같이 다음과 같은 종류를 가질 수 있다. (envelope, small box, large box)

Weight: 해당 패키지의 무게를 뜻한다.

Estimated_date: 도착 예정 날짜를 저장하는 속성이다.

Hazardous: 위험 물품인지를 나타내는 속성이다.

International: 해외로 배송되는 제품인지를 나타내는 속성이다.

4) Bill

package마다 청구서, 결제서가 발생한다고 가정하였다. 그렇기 때문에 package와 one-to-one의 관계를 가지며 customer와는 many to one의 관계를 갖는다. 이는 한 명의 고객이 여러 개의 청구서를 가질 수 있지만 청구서는 각각 1명의 고객과 연결되기 때문이다.

package_ID(FK): package로부터 받아온 package_ID가 Bill entity의 primary key가 된다. 앞서 package마다 청구서가 발생한다고 가정하였기 때문이다. Package_ID를 foreign key로 받기 때문에 Bill entity는 package entity에 대해 weak entity로 이해될 수 있다.

Customer_ID(FK): 청구서에 찍힌 비용을 지불할 고객의 정보를 가리키는 속성이다. Customer entity의 primary key를 참조하기 때문에 foreign key이다.

Charge: 각 패키지에 대해 청구되는 비용을 저장하는 속성이다.

Due_date: 청구된 금액을 결제해야 하는 기한 날짜를 저장하는 속성이다.

5) Delivery_Info

package가 어떤 경로로 운송되고 있는지, 그리고 현재 어떤 상태에 있는지를 추적하기 위해 만들어진 entity이다. 정보의 중복이 일부 발생할 수 있지만 명세서에서 배달 업체는 배송되는 과정의 세부 사항들을 추적하고 있어야 한다고 했기 때문에 따로 entity로 만들었다. 또한 package, location, vehicle의 primary key를 foreign key로 받아서 자신의 primary key를 정의하기 때문에 세 entity에 대해 모두 weak entitiy라고 볼 수 있다.

굳이 네 가지 속성을 primary key로 갖는 이유는 delivery info에 정보를 저장할 때 package_Id 만으로 식별할 수 없기 때문이다. 명세서에서 배송 과정에 대한 세부정보를 가져야 한다고 했기 때문에 delivery info에 저장되는 정보는 package 별로 1개씩만 있지 않다. 즉, 같은 차량이 같은 패키지를 담고 장소를 이동하기 때문에 package_id 만으로는 delivery_info에 저장되는 튜플들이 유일하게 식별되지 않는다. 같은 지역을 두 번 지나가는 경우도 있기 때문에 cur_location에 추가로 heading_to도 추가하여 총 네 가지의 속성(package_id, cur_location, heading_to, vehicle_id)의 조합으로 delivery_info의 튜플을 유일하게 식별할 수 있다.

Package_ID(FK): package entity에서 가져온 정보이다.

Cur_location(FK): location entity에서 location_name을 cur_location이라는 이름으로 가져왔으며 현재 package가 어디에 위치해 있는지를 나타낸다.

Heading_to(FK): location entity에서 location_name을 heading_to라는 이름으로 가져왔으며 현재 배송되고 있는 package가 다음으로 가고 있는 장소를 의미한다.

Vehicle_ID(FK): transportation entity에서 가져온 정보로 현재 package가 배송되고 있는 운송 수단의 정보를 저장한다.

Cur_time: 각 package에 대해 현재 위치가 어디인지를 저장할 때 그 시간대를 저장하기 위한 정보이다.

Delivery_status: 배달 상태를 저장한다. 예를 들어, (store, move, delay, crash, done)으로 저장되는데 store은 아직 warehouse에 저장되어 있다는 뜻이고 move는 운송 수단을 갖고 이동 중이라는 것을, delay는 운송 중이지만 도착 시간보다 늦었음을 의미하고, crash는 운송 중에 예기치 않은 사고의 발생으로 배송할 수 없는 상태를 의미하며 마지막으로 done은 배송이 완료되었음을 나타낸다.

6) Transportation

Delivery_Info에서 package가 배달되는 경우 어떤 운송 수단을 이용하고 있는지 정보를 나타내고자 할 때 참조하기 위해 만들어진 entity이다.

Vehicle_ID: 운송 수단의 고유한 일련번호를 저장하기 때문에 primary key로 간주된다.

Vehicle_type: 운송 수단마다 고유한 번호가 있지만 각 수단의 종류는 구분된다. 예를 들어 truck인지, plane인지 등이 저장될 수 있다.

7) Location

Location은 Delivery_Info에서 현재 package가 어디 위치해 있는지를 나타내고자 할 때 참조하기 위해 만든 엔티티이다. 너무 자세한 주소를 저장하기 보다는 특정 도시에 있다는 정보만 저장하면 되기 때문에 많은 tuple이 저장되지는 않을 것으로 예상된다.

Location_name: 유일한 속성이자 primary key로 지역의 이름이 곧 지역의 주소를 의미한다고 가정한다. 물론 현실에서는 이름이 동일한 지명도 있지만 앞서 이야기했듯 도시 단위로 저장하기 때문에 우리나라를 기준으로 보았을 때 중복된 tuple은 발생하지 않을 것으로 예상되어 이렇게 속성을 설정하였다.

1.2 Relationship Set

1) payment(customer – bill)

Customer와 bill 사이의 relationship으로 청구서에 나온 금액을 결제해야 하는 상황에서 발생하는 속성들을 추가적으로 갖고 있다. 속성을 살펴보기 이전에 customer와 bill의 관계를 보면 고객 한 명당 여러 개의 청구서를 가질 수 있는 반면 청구서는 한 명의 고객과만 연결되어야 한다. 따라서 one-to-many의 cardinality를 가지며 청구서가 없는 고객은 있어도 청구서는 무조건 특정 고객과 연결되어야 하기 때문에 many쪽이 total이 되어야 한다.

payment_method: 결제가 어떤 방식으로 되는 지를 나타내는 속성으로 신용 카드, 이체, 현금 등의 값을 가질 수 있다.

payment_status: 결제 상태를 나타내는 속성으로 미결제, 미리 결제(prepaid), 매월 결제(monthly) 등을 저장한다.

payment_account: 결제에 사용된 계좌번호를 저장한다

payment_date: 결제가 완료되었다면 결제가 된 날짜 정보를 저장하는 속성이다.

2) order(customer – package)

한 명의 고객이 여러 패키지를 주문할 수 있지만 한 개의 패키지는 한 명의 고객하고 관계하기 때문에 one-to-many 관계이며 모든 package는 항상 customer와 관계해야 하기 때문에 many쪽이 total로 설정되어야 한다.

Recipient_name: 주문을 했을 때 수령인의 이름을 저장한다.

Recipient_address: 주문을 했을 때, 수령인의 주소를 저장한다.

Recipient_phone_num: 주문을 했을 때, 수령인의 전화번호를 저장한다.

Order_time: 주문 했을 때의 시간을 저장한다.

3) billing (package – bill)

Package와 bill 사이의 relationship이며 one to one 관계를 갖는다. 앞서 bill entity에서 설명했듯 하나의 패키지마다 하나의 청구서가 발생한다는 가정을 했기 때문이다. 또한 bill entity는 pacakge의 primary key인 package_ID를 foreign key로 갖으며 자신의 primary key로 갖기 때문에 package entity에 의존하게 되어 weak entity로 생각할 수 있다.

4) contract (customer – shipper)

Customer와 shipper 사이의 관계로 명세서에서 언급되었던 사항을 반영하기 위해 도입된 관계이다. ("Some customers have a contract with the shipper") 계약한 고객이 있는 반면 없을 수도 있으니 partial이고 고객은 한 명의 shipper와 관계하고 shipper는 여러 명의 고객과 관계할 수 있으므로 many-to-one의 관계를 갖는다.

5) tracking (package – delivery_info)

패키지와 delivery_info는 1대 1로 관계한다. 그러므로 one-to-one 관계인데 delivery_info가 package_id를 foreign key로 가져 package entity에 의존하기 때문에 delivery_info entity는 weak entity이고 package entity는 strong entity로 볼 수 있다.

6) vehicle (delivery info – transportation)

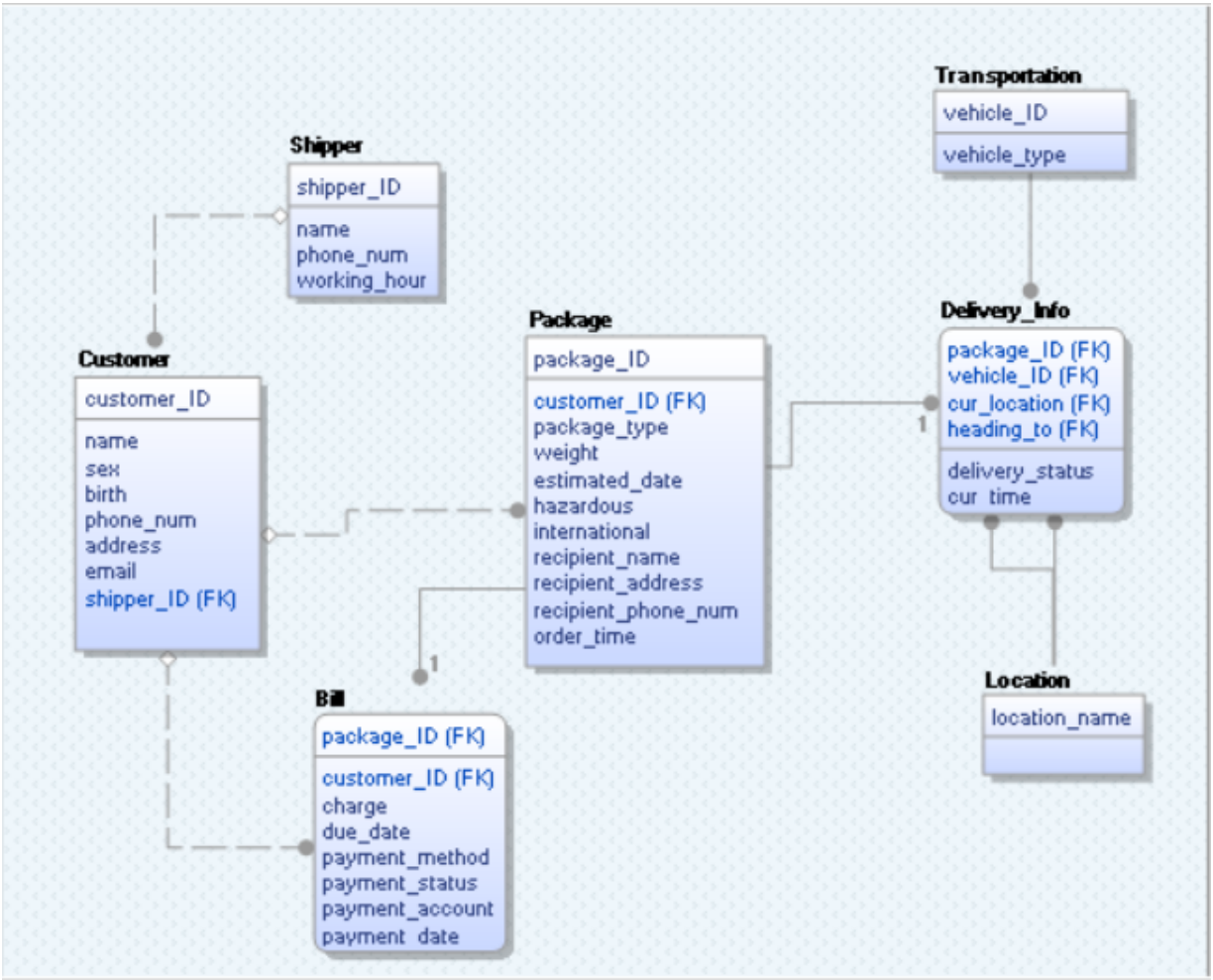
각각의 Delivery_info는 하나의 운송 수단과 관계하고 한 종류의 운송 수단은 여러 패키지의 배달 정보와 관계할 수 있기 때문에 many-to-one의 관계가 성립된다. 또한 delivery info는 vehicle_ID

를 foreign key로 가지며 동시에 primary key의 요소 중 하나로 갖는다. 따라서 delivery info는 transportation에 의존하기 때문에 transportation entity에 대해 weak entitiy라 볼 수 있다.

7) position (delivery info – location)

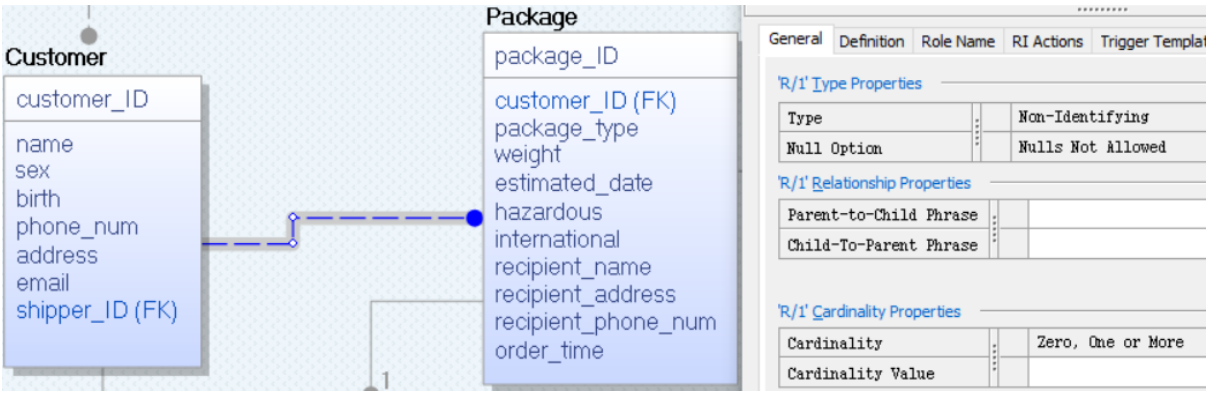
각각의 Delivery_info는 한 장소와 관계하고 한 장소는 여러 패키지의 배달 정보와 관계할 수 있기 때문에 many-to-one의 관계가 성립된다. 또한 delivery info는 location_name을 foreign key로 가지며 동시에 primary key의 요소 중 하나로 갖는다. 따라서 delivery info는 location에 의존하기 때문에 location entity에 대해 weak entitiy라 볼 수 있다.

2.1 Relation Schema Diagram



2.2 Relation Schema

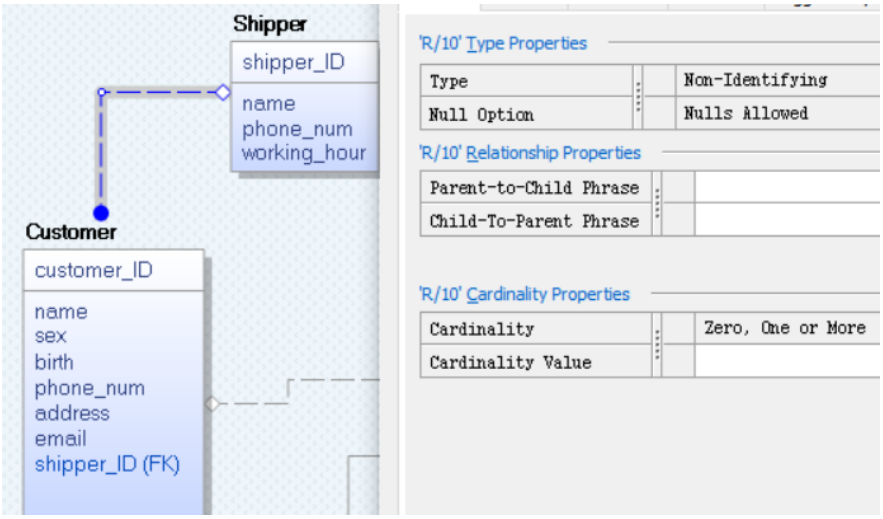
1) Customer - Package



Customer - Package에서는 ER Model에서 'order'이라는 relationship set이 존재했지만 customer-package가 one-to-many 관계이며 many쪽이 total이기 때문에 many쪽에 속성들을 추가시킴으로써 relationship set을 생략할 수 있다. 우선, one쪽의 primary key를 Foreign key로 넣고 order의 attribute(recipient_name, recipient_address, recipient_phone_num, order_time)를 many 쪽에 넣음으로써 relationship set을 생략했다.

Cardinality를 살펴보면, customer는 package를 0~many까지 가질 수 있고 package는 customer를 무조건 1명 가져야 하기 때문에 one-to zero, one or more이 된다. 그리고 total이기 때문에 Nulls Not Allowed로 설정했다.

2) Shipper – Customer

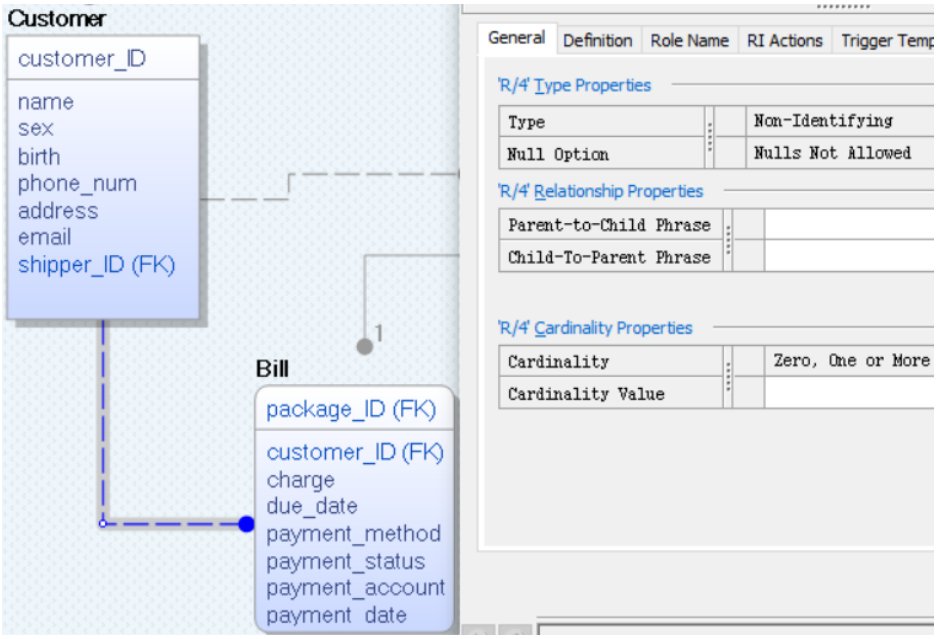


E-R Model에서는 shipper와 customer의 relationship set으로 contract가 있었지만 one쪽(shipper)의 primary key(shipper_ID)를 many쪽에 넣음으로써 relationship set을 생략할 수 있다. 그렇기 때

문에 shipper_ID가 foreign key로 customer에 추가되었음을 확인할 수 있다.

Cardinality를 살펴보면, shipper – customer는 one-to-many의 관계이며 shipper는 customer를 0~many를 가질 수 있으며 customer는 0~1의 shipper를 가질 수 있다. 따라서 zero, one or more이 들어가야 한다. 그리고 total이 아니기 때문에 Nulls allowed로 설정했다.

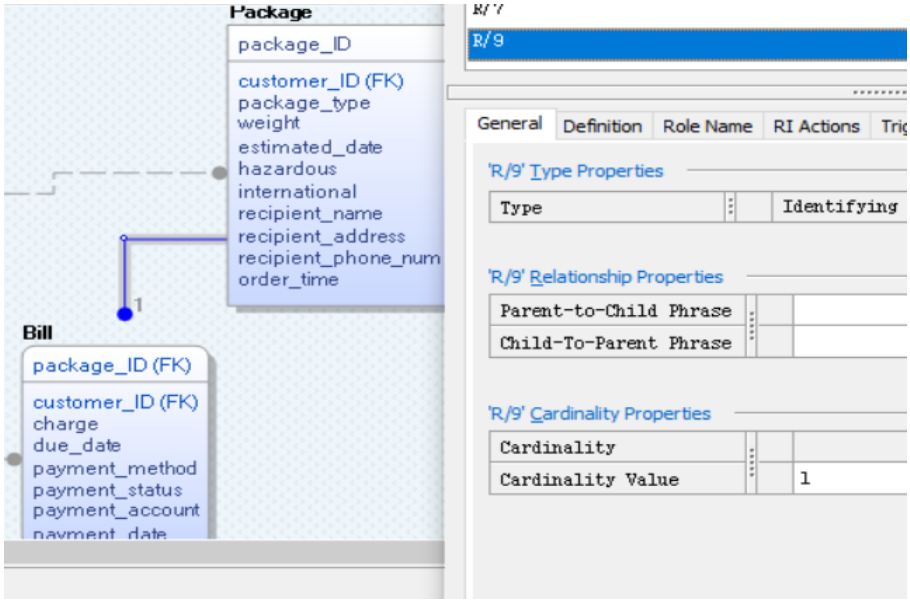
3) Customer – Bill



E-R Model에서 Customer – bill은 one-to-many의 관계이며 relationship set으로 payment가 있었다. Relation Schema로 만들 때는 다음과 같은 조치를 취함으로써 relationship set을 생략할 수 있다. 우선, one쪽에 해당하는 customer의 primary key를 many쪽인 bill에 foreign key로 추가하며 relationship set이었던 payment가 갖고 있던 속성들을 many쪽인 bill에 추가함으로써 payment를 생략할 수 있다.

Cardinality를 살펴보면, customer는 bill을 0~many개 가질 수 있지만 bill은 1개의 고객만을 가질 수 있다. 따라서 one-to-zero, one or more을 선택한 것을 확인할 수 있다. 그리고 total이기 때문에 Nulls Not Allowed로 설정했다.

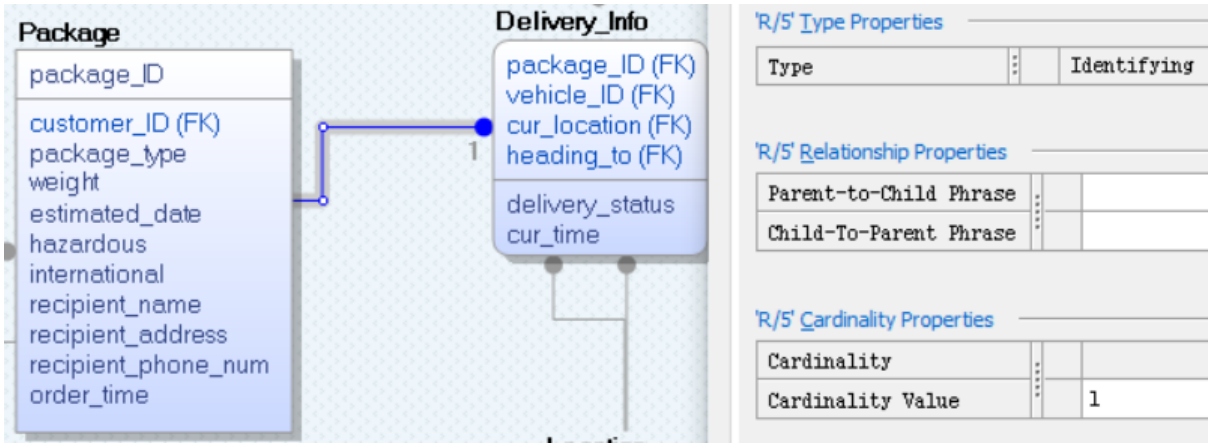
4) Package – Bill



Package – bill은 one-to-one의 관계이다. 따라서 Cardinality에 1을 기입하여 이를 표현하였다.

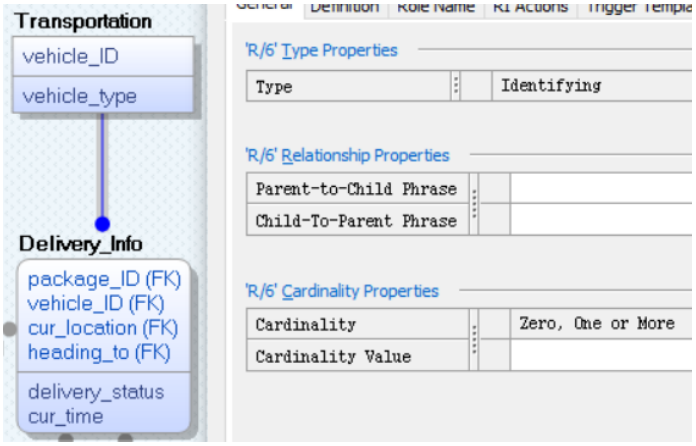
또한 Bill은 package의 primary key로 정의되는 Weak Entity이기 때문에 'Identifying'으로 표시되도록 설정하였다.

5) Package – Delivery_Info



E-R Model에서 package-delivery_info는 one-to-one의 관계를 가졌으며 delivery_info가 package_ID를 foreign key로 가져와 primary key의 일부로 사용하기 때문에 delivery_info는 package에 대해 weak entity였다. Cardinality는 one-to-one이기 때문에 cardinality value에 1을 설정하였다.

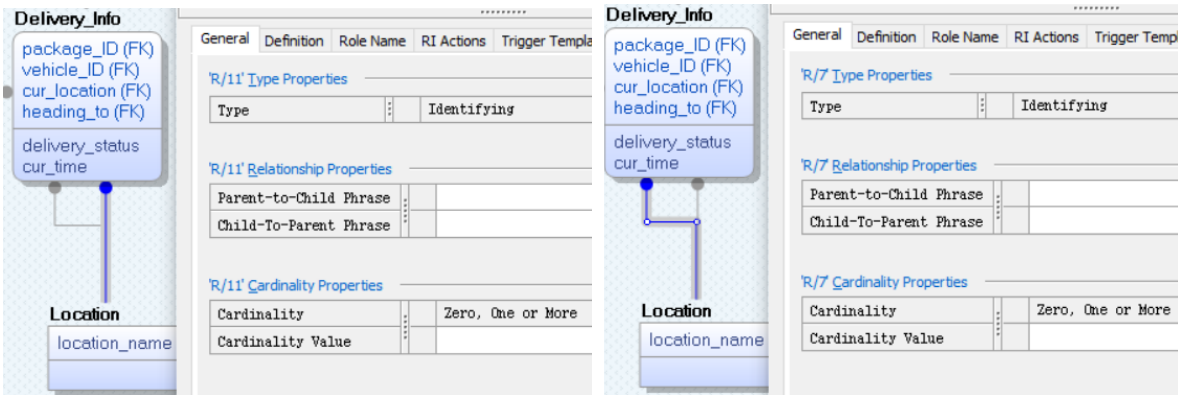
6) Delivery_Info – Transportation



Delivery_Info는 Transportation의 primary key에 의존하기 때문에 Weak entity이다. 따라서 vehicle_ID가 foreign key로 Delivery_Info의 primary key를 구성하고 있으며 identifying으로 타입이 설정되어 있다.

Cardinality를 살펴보면, Transportation 1개가 Delivery_Info의 튜플에 하나도 적용되지 않거나 하나 혹은 여러 개에 적용될 수 있기 때문에 one-to-zero, one or more로 설정되었다.

7) Delivery_Info – Location



Delivery_Info는 Location의 primary key(location_name)에 의존하기 때문에 Weak entity이다. 특이한 점은 foreign key로 가져올 때 cur_location과 heading_to라는 두 가지 속성으로 참조해온다는 것이다. Cur_location은 현재 위치, heading_to는 다음으로 향하는 위치를 저장한다. 결국 이 역시 location의 primary key가 foreign key로 Delivery_Info의 primary key를 구성하고 있으므로 identifying으로 타입이 설정되어 있다.

Cardinality를 살펴보면, Location 1개가 Delivery_Info의 튜플에 하나도 적용되지 않거나 하나 혹은 여러 개에 적용될 수 있기 때문에 one-to-zero, one or more로 설정되었다.

3. Query

1) 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.

1번 query는 truck 1721이 사고가 난 상황에서 해당 사고에 포함된 패키지의 고객을 출력하라는 것, 수령인을 출력하라는 것, 그리고 같은 트럭에서 마지막으로 성공적으로 배송된 것을 찾으라는 3가지 query로 구분해서 생각할 수 있다.

우선 첫 번째 고객을 찾는 것은 다음과 같은 query를 통해서 가능하다

```
Select customer_ID
From Package, (Select distinct package_ID
From Delivery_Info
Where vehicle_ID = '1721' and delivery_status = 'crash') as A
Where Package.package_ID = A.package_ID
```

두 번째 수령인을 찾는 것도 유사한 원리로 수행되며 다음과 같은 query를 통해서 가능하다

```
Select recipient_name
From Package, (Select distinct package_ID
From Delivery_Info
Where vehicle_ID = '1721' and delivery_status = 'crash') as A
Where Package.package_ID = A.package_ID
```

세 번째로 마지막으로 성공된 배달을 찾으라는 query는 다음을 통해서 가능하다.

```
select package_ID
from Delivery_Info
where vehicle_ID = '1721'
        and delivery_stauts = 'done'
```

and max(cur_time)

2) Find the customer who has shipped the most packages in the past year.

2번 query를 해결하기 위해서는 bill entity를 활용하면 가능하다. 다음과 같은 query가 가능하다.

select A.customer_ID

from bill A, bill B

group by customer_ID

having count(A.package_ID) > count(B.package_ID)

and A.due_date like "2022%"

and B.due_date like "2022%"

3) Find the customer who has spent the most money on shipping in the past year.

3번 query는 2번 query와 유사하지만 money를 계산하는 것이 달라졌다. 다음과 같은 query가 가능하다.

select A.customer_ID

from bill A, bill B

group by customer_ID

having sum(A.charge) > sum(B.charge)

and A.due_date like "2022%"

and B.due_date like "2022%"

4) Find those packages that were not delivered within the promised time.

예정된 시간에 배달되지 못했다는 것은 배달이 delay가 되었다는 것이고 이는 Delivery_Info의 속성 중 하나인 delivery_status를 살펴봄으로써 쉽게 찾아낼 수 있다. 단, delivery info에서는 package_ID가 중복된 tuple이 있으므로 distinct를 사용해서 중복을 없애고 출력해내야 한다. 이를 반영하면 다음과 같은 query가 가능하다.

select distinct package_ID

from Delivery_Info

where delivery_status = 'delay'

5) 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.

우선 간단한 bill은 다음과 같은 query를 통해서 찾아낼 수 있다.

select customer_name, customer_address, sum(charge) as amount

from Customer, (select customer_ID, sum(charge)

from Bill

group by customer_ID) as A

where Customer.customer_ID = A.customer_ID

이후의 query들 역시 bill과 customer entity를 활용하여 구현할 수 있을 것으로 예상된다.