DB PROJECT1 REPORT

20200901 이효주

<1> Analysis

주어진 명세서에 따르면, Package delivery company의 DBA로서 Package handling과 billing 측면에서 데이터베이스를 구축해야한다.

아래는 명세서에서 고려해야 할 조건들을 분석하였다.

1. 여러 종류의 서비스가 존재할 수 있다. package의 type(flat envelope, small box, larger boxes), package의 weight, 그리고 delivery timeliness(overnight, second day, or longer)등에 대한 데이터를 고려해야 한다.

   ➔ 각 package마다 service type이 다를 수 있으므로, package라는 entity set에 속성으로 weight, type, timeliness를 추가하였다.

2. 일부 고객은 매달 계좌 번호로 금액을 청구하고, 자주 이용하지 않는 다른 고객들은 신용카드 결제를 이용한다.

   ➔ Regular_customer와 Infrequent_customer라는 entity를 설정하여 따로 정보를 저장할 수 있게 설정하였다. 그리고 Regular_customer의 속성에는 account_number를 추가하였고, Infrequent_customer의 속성에는 credit_card_number를 추가하였다.

   ➔ 회사와 계약을 맺은 customer이 매달 bill을 받지만, 계약을 하지 않은 customer도 한 달 동안의 주문에 대한 영수증 개념의 bill을 요구할 수 있다고 생각하여, bill이라는 entity를 따로 설정하여 customer과 관계를 맺도록 하였다.

3. 대부분의 경우 shipping 회사는 무엇이 선적되는지 신경 쓸 필요가 없으나, 위험물질 혹은 내용물과 value를 명시한 세관 신고가 필요한 국제 발송에 대한 정보는 고려할 수 있어야 한다.

   ➔ 각 package마다 내용물과 내용물에 대한 가치, 위험물질 등을 알 수 있도록 package entity에 content_value, content_description에 대한 속성을 추가하였다.

4. 회사는 고객이 package를 내려준 시점(또는 회사가 픽업한 시점)부터 배송 및 서명할 때까지 패키지를 추적해야 한다. Package가 어디에 있는지, 어디에 있었는지, 현재 어디로 향하고 있는지에 대한 모든 세부사항을 알 수 있어야 한다. 그리고 회사는 언제든지 package가 어떤 트럭, 비행기 혹은 창고에 있는지도 알아야 한다.

➔ Entity와 relation을 짜면서 이 조건을 가장 크게 고려하였다. Package의 출발 시간, 도착 시간, 도착 예정시간에 관한 정보는 recipient entity의 속성으로 추가하였다. 그리고 delivery_status라는 속성도 추가하여 recipient에게 package가 운송전인지, 운송 중 인지, 운송 완료되었는지 알 수 있도록 하였다.

➔ 보관의 측면에서는, package entity와 shipment entity를 warehouse라는 entity를 생성해 관계를 만들었다. 어떤 shipment, 그리고 개별로 어떤 package가 어느 창고에 있었는지 알 수 있도록 관계를 형성하였다. Package가 어느 창고에 보관되어 있다가, 한 shipment에 속해 배송이 되던 중 다시 창고에 보관될 수 있는 가능성을 생각하여서 package와 shipment 각각 warehouse와 관계를 맺을 수 있도록 하였다.

➔ Tracking의 시점에서는 먼저, current_shipping_traking이라는 entity를 만들어, 현재 배송되고 있는 shipment를 추적할 수 있도록 하였다. 현재 위치, 현재 운송수단, 향하고 있는 위치, 운송 상태를 속성으로 하였다. 만약 운송 상태가 현재 배송 중이 아니라면 current_shipping_traking에서 history_of_shipment_traking이라는 entity에 저장될 것이다. 이것은 한 shipment 당 여러 데이터가 존재할 수 있도록 하였고, 과거 시점에서의 위치, 시간, 운송수단 등을 확인할 수 있도록 하였다.

➔ 만약 운송이 완료된 후, 마지막 운송 수단을 찾고싶다면 history_of_shipment_ trakcing entity에서 가장 최근 시점의 transportation_id 속성을 찾으면 될 것이다.

<2> E-R diagram

위의 명세서의 조건들을 고려하여 고객 및 배송 관련 데이터를 포함한 E-R diagram을 구성하였다.

먼저 entity sets은 고객(Customer), 비정기적 고객(Infrequent customer), 정기적 고객(Regular customer), 청구서(Bill), 주문(Order), 창고(Warehouse), 소포(Package), 수신인(Recipient), 출하(Shipment), 현재 배송 추적(Current shipping tracking), 운송(Transportation), 배송 추적 기록(History of shipment tracking)로 설정하였다.

Entity**와 그 속성**에 대해 자세히 설명해보면,

Customer:

customer_id(PK): 고객 ID

phone_number: 전화번호

address: 주소

email: 이메일

name: 이름

Infrequent customer:

regular_customer_id(PK): 자주 이용하는 고객 ID

credit_card_number: 신용카드 번호

Regular customer:

infrequent_customer_id(PK): 가끔 이용하는 고객 ID

account_number: 계좌 번호

//여기서 Infrequent customer과 Regular customer의 entity를 따로 생성하여, 계좌번호로 결제하는 고객과 신용카드를 결제하는 고객의 데이터를 따로 저장하였다. Customer entity 한 개만 생성할 경우, 계좌번호를 알고 있을 때 credit card가 NULL이 될 수도 있으므로 NULL값을 피하면서 회사와 계약을 맺은 고객에 대한 정보만 보고싶을 경우 빠른 검색을 위해 다른 entity set으로 저장하였다.

Bill:

bill_id(PK): 청구서 ID

customer_id(FK): 고객 ID

total_payment_amount: 총 지불 금액

payment_date: 지불 날짜

payment_status: 지불 상태

billing_date: 청구 날짜

Order:

order_id(PK): 주문 ID

customer_id(FK): 고객 ID

package_id(FK): 소포 ID

order_price: 주문 가격

payment_status: 지불 상태

payment_date: 지불 날짜

Warehouse:

warehouse_id(PK): 창고 ID

location: 위치

Package:

package_id(PK): 소포 ID

content_value: 내용물 가치

content_description: 내용물 설명

timeliness_of_delivery: 배송 시간

package_weight: 소포 무게

package_type: 소포 유형

package_origin: 소포 출발지

package_destination: 소포 목적지

Recipient:

recipient_id(PK): 수령인 ID

customer_id(FK): 고객 ID

package_id(FK): 소포 ID

recipient_address: 수령인 주소

delivery_completion_time: 배송 완료 시간

delivery_departure_time: 배송 출발 시간(package 픽업 시간)

promised_delivery_time: 약속된 배송 시간

delivery_status: 배송 상태

//Recipient entity에 배송 완료 시간, 배송 출발 시간, 약속된 배송시간, 배송 상태를 속성값으로 추가하여서 지금 package가 어디에 있는지, 언제 도착했는지 등을 세세하게 추적할 수 있도록 하였다. 만약 약속된 배송 시간 보다 늦게 도착했다면 이 entity에서 확인할 수 있을 것이다.

//그리고 package에는 package_destination이 있고, recipient에는 recipient_address가 존재하는데, 소포 목적지와 수령인의 주소가 다를 수 있으므로 중복이 아니라 생각하고 따로 설정하였다. 그리고 recipient entity를 customer entity와 구별하여서 따로 설정한 이유는, 수령인 중 우리 고객이 있을 수도 있지만, 우리 고객이 아닌 경우도 많을 것이라고 생각하기 때문에 따로 recipient라는 entity set을 설정하고, customer와 관계를 설정하였다.

Shipment:

shipment_id(PK): 운송 ID

shipment_origin: 출발지

shipment_destination: 목적지

shipment_date: 운송 날짜

// 여기서도 마찬가지로 shipment_origin과 shipment_destination과 package_origin, package_destination을 따로 설정한 경우는 shipment의 경우 바로 package_destination이 아닌 창고나 다른 지역으로 이동할 수 있고, 이동 뒤 다른 shipment와 합쳐질 수 있다고 생각했기 때문에 shipment의 출발지와 도착지를 다르게 설정하였다.

Current_Shipping_Tracking:

tracking_id(PK): 배송 추적 ID

shipment_id(FK): 운송 ID

current_location: 현재 위치

current_location_time: 현재 위치 시간

transportation_id(FK): 운송수단 ID

currently_headed_location: 현재 향하는 위치

delivery_status: 배송 상태

History_of_Shipment_Tracking:

tracking_history_id(PK): 배송 추적 이력 ID

shipment_id(FK): 운송 ID

location: 과거 위치

time: 과거 시각

transportation_id(FK): 과거 운송수단

// Current_shipping_Traking entity에 대해 추가적인 설명을 하자면, 배송 상태가 현재 운송 중인 shipment를 추적하는 entity로 설정하였다. 따라서 현재 배송 중인 shipment의 현재 위치와 그에 따른 시간, 현재 운송 수단, 그리고 현재 향하는 곳에 대한 정보를 알 수 있다. 만약 배송이 완료되었거나, 일정 시간이 지나면 History_of_shipment_tracking에 한 데이터로서 저장될 것이다. 이렇게 하여 언제 어느 시점에 어떤 운송수단이 사용되었고, 위치가 어디였는지 추적할 수 있을 것이다. 그렇다면, History_of_shipment_tracking entity는 한 shipment의 trakcing에 대해서는 여러 history가 존재할 수 있고, time 속성을 기준으로 가장 최근 시점의 운송수단 등을 확인할 수도 있을 것이다.

Transportation:

transportation_id(PK): 운송수단 ID

transportation_number: 운송수단 번호

transportation_method_type: 운송수단 유형

driver_id: 운전자 ID

driver_name: 운전자 이름

driver_phone_number: 운전자 전화번호

Entity**들 간의 관계**를 설명해보면,

1. Customer**과** Regular_customer**은** 1(partial):1(total)**의** have_a_contract **관계**

- 고객(Customer)와 자주 이용하는 고객(Regular_customer)은 서로 one to one의 관계를 가진다. Regular_ customer은 '계약'에 의서 계좌 번호로 결제를 진행한다. Regular customer은 반드시 Customer entity에 한 명으로 존재해야 하기 때문에, total 관계를 표시하였다.

2. Customer**과** Infrequent_customer**은** 1(partial):1(total)**의** infrequent_customer_paying_with_ creditcard **관계**

• 고객(Customer)과 가끔 이용하는 고객(Infrequent_customer)은 서로 one to one의 관계를 가진다. Infrequent_customer는 신용카드를 이용하여 결제를 진행하고, Infrequent customer은 반드시 Customer entity에 한 명으로 존재해야 하기 때문에, total 관계를 표시하였다.

3. Customer**과** Recipient**는** 1(partial):1(partial)**의** customer_recipeint **관계**

- 고객(Customer)과 수령인(Recipient)은 서로 one to one의 관계를 가진다. 이 관계에서 소포(Package)의 배송 상황을 파악할 수 있다. 그리고 total이 아닌 이유는, recipient 중 회사 고객이 아닌 수령인이 존재하고, 고객이 모두 수령인이 되는 것은 아니기 때문에 partial 관계로 설정하였다.

4. Customer**과** Order**은** 1(partial):N(total)**의** customer_order **관계**

- 고객(Customer)과 주문(Order)은 서로 one to many의 관계를 가진다 즉, 하나의 고객이 여러 개의 주문을 할 수 있고, 하나의 주문은 하나의 고객에게만 속해야한다. 이 관계를 통해서 어떤 고객이 어떤 주문을 했는지 알 수 있고, 고객의 총 금액과 결제 상태도 알 수 있을 것이다. 그리고 order은 반드시 customer이 주문을 해야 데이터가 생성되기 때문에 반드시 이 관계에 참여해야 하므로 total으로 설정하였고, 모든 고객이 주문을 했다고 보기에는 주문을 했다가 바로 취소 한 경우, 그리고 계약만 하고 아직 주문을 하지 않은 경우 등 수많은 경우가 존재할 것 같아서 partial로 설정하였다.

5. Customer**과** Bill**은** 1(partial):N(total)**의** payment **관계**

- 고객(Customer)과 청구서(Bill)은 서로 one to many의 관계를 가진다. 즉, 하나의 고객은 여러 개의 청구서를 받을 수 있다. 청구서는 monthly로 청구되기 때문에 시간에 따라 여러 청구서를 가질 수 있을 것이다. 그리고 Infrequent_customer은 청구서를 받지 않기 때문에, Regular_customer와 관계를 형성할 수 있기는 하였으나, customer entity와 관계를 형성한 이유는, 종종 사용하던 고객이 계약을 통해 regular_customer가 될 수 있고, bill을 받지 않더라도, 고객의 요청에 따라 이번 달의 주문 금액을 영수증 용도로 요구할 수 있기 때문에 customer entity와 관계를 형성하였다. 그리고 bill은 무조건 customer과 관계를 형성해야 하므로 total 관계를 설정해주었다. 모든 고객이 bill을 받는 것은 아니므로 customer은 partial 관계를 설정 해 주었다.

6. Order**과** Package**는** 1(total):1(total)**의** order_a_package **관계**

- 주문(Order)과 소포(Package)는 서로 one to one의 관계를 가진다. 즉, 하나의 주문은 하나의 소포와 연결될 것이다. 그리고 모두 total 관계로 설정하였는데, 주문과 package는 반드시 1 대 1 관계로 참여해야 하기 때문이다. 한 사람이 여러 개의 package를 한 번 주문하였더라도, 도착지나 type 등이

다를 수 있으므로 한 package 당 하나의 주문으로 생각하고 관계를 설정하였다. 만약 한 사람이 10 개의 package를 한 번에 주문했다 하더라도, 한 package씩 열 번의 주문이 들어왔다고 생각하는 것 이다.

7. Package**와** Recipient**는** N(total):1(total)**의** delivery_to_recipient **관계**

- 소포(Package)와 수령인(Recipient)은 many to one의 관계를 가진다. 여러 개의 소포가 하나의 수령인 에게 배송될 수 있다. 다른 customer가 우연히 한 명의 recipient에게 package를 보낼 수 있고, 한 고객이 수령인에게 여러 번 보낼 수 있기 때문에 many to one의 관계로 설정하였다. 그리고 package가 존재하면 수령인이 존재해야하고, 수령인이 있으면 그에 맞는 package가 존재해야 할 것 이기 때문에 모두 total로 설정하였다.

8. Package**와** Warehouse**는** N(partial):M(partial)**의** storage_of_package **관계**

- 소포(Package)와 창고(Warehouse)는 many to many의 관계를 가진다. 즉, 여러 개의 소포가 여러 개 의 창고에 보관될 수 있다. Package가 한 창고에만 보관되는 것이 아니고, 운송 되어 또 다른 창고 에 보관되었다가 옮겨질 수 있다고 생각했기 때문에 many to many로 설정하였고, 빈 창고도 존재할 수 있고, 아직 창고에 보관되지 않은 package나, 보관되지 않고 바로 배송되는 package도 존재할 것이기 때문에 둘 다 partial로 설정하였다.

9. Warehouse**와** Shipment**는** N(partial):M(partial)**의** storage_of_shipment **관계**

- 창고(Warehouse)와 운송(shipment)는 many to many의 관계를 가진다. 여러 개의 package들의 집합인 shipment가 여러 개의 창고에 보관될 수 있다. 위와 마찬가지로 shipment가 여러 창고에 보관될 수 있고, 창고에는 여러 shipment가 보관될 수 있기 때문이다. 또한 빈 창고도 존재할 수 있고, 아직 창고에 보관되지 않은 shipment나, 보관되지 않고 바로 배송되는 shipment도 존재할 것이기 때문에 둘 다 partial로 설정하였다.
- 추가적으로, package와 warehouse, shipment와 warehouse의 관계를 따로 설정한 이유는, 한 package 가 창고에 보관된 이후에 다른 창고로 옮겨져 분리 과정을 거친 후 shipment로 운송될 수 있기 때 문에 반드시 한 package와 그 package가 포함된 shipment가 반드시 동일한 창고에 있었다고 할 수 없다. 따라서 package와 shipment 따로 warehouse와 관계를 설정하였다.

10. Package**와** Shipment**는** N(total):1(total)**의** shipment_package **관계**

- 소포(Package)와 소포의 집합인 shipment는 many to one의 관계를 가진다. 여러 개의 소포가 하나의 shipment로 구성될 수 있기 때문이다. 그리고 모두 total로 설정하였는데, package는 반드시 shipment에 포함되어야 하고, shipment는 반드시 package에 포함되어야 한다. 운송 회사이기 때문에 퀵 서비스와 같이 하나의 pakage를 수령인에게 바로 보내주는 경우는 없을 것이고, 적어도 같은

목적지의 두 개의 package가 모여 하나의 shipment를 구성할 것이라 생각하였기 때문에 total 관계로 설정하였다.

11. Current_shipping_Traking**과** History_of_shipment_tracking**는** 1(partial):N(partial)**의** History_of_shipment_trakcing **관계**

• 운송 중인 현재 상태(current_shipping_trakcing)와 이전 상태(history_of_shipment_tracking)는 one to many의 관계를 가지며, 하나의 운송 중인 소포는 여러 개의 history를 가질 수 있기에 이렇게 설정하였다. 지금 막 배송이 시작되어서 이전 상태에 대한 정보가 없거나, 배송이 완료되거나 시작되지 않아서 현재 상태에 대한 정보가 없을 수 있으므로 partial 관계로 설정하였다. 조금 더 자세하게 설명하자면, 현재 운송중인 shipment에서 추적하고, 특정 시간이 지난 후에는 history_of_shipment_tracking에 저장되어 과거 정보로 과거의 위치와, 시간, 운송수단 등에 대한 데이터 뿐만 아니라 가장 최근에 배송을 완료한 shipment 등의 정보도 읽어올 수 있을 것이다.

12. Current_shipment_tracking**과** Transportation**은** N(total):1(partial)**의** tracking_with_transportation **관계**

• 운송 중인 소포(Shipment)의 운송 상태(current_shipment_tracking)는 특정 운송 수단(Transportation)에 의해 이루어진다. 현재 운송중인 여러 개의 shipment가 하나의 운송 수단에 의해 운송될 수 있기 때문에 many to one관계로 설정하였다. 예를 들면 여러 개의 shipment가 비행기로 운송될 수 있기 때문이다. 그리고 현재 운송 중인 shipment는 반드시 운송 수단을 가져야 하므로 total으로 설정하였고, 반드시 모든 운송수단이 현재 운송을 하고 있을 필요는 없으므로 partial로 설정하였다.

13. History_of_shipment_tracking**와** Transportation**은** 1(total):1(partial)**의** tracking_history_with_transportation **관계**

- 운송 중인 소포의 과거 상태(History_of_shipment_tracking)와 운송 수단(Transportation)은 one to one 관계를 지닌다. 하나의 shipment에 여러 과거 상태가 존재할 수 있고, 그 과거 상태 중 하나의 상황은 오직 하나의 운송수단을 이용하고 있어야 할 것이다. 예를 들면 비행기 -> 트럭 이렇게 옮겨갔다면 하나의 shipment에 두 가지의 history가 존재하는 것이다. 그리고 만약 과거 운송 중인 소포의 상황에 대한 데이터가 존재한다면, 반드시 운송수단도 존재해야 하므로 total로 설정하였고, 모든 운송수단이 history_of_shipment_tracking에 참여할 필요는 없기에 이 부분은 partial로 설정하였다.

14. Current_shipment_tracking**과** Shipment**의** 1(total):1(partial)**의** tracking_of_shipment **관계**

- 운송 중인 현재 상태(current_shipping_trakcing)와 shipment는 one to one의 관계를 지닌다. 만약 shipment가 현재 운송 중이라면, '현재 시점'에 대한 한 가지의 정보만 가지고 있어야 할 것이다. 그리고 모든 shipment가 현재 운송중인 상황은 존재할 수 없으므로 partial로 설정하였고, 운송 중인 shipment_tracking이 있다면 shipment와의 관계에 참여해야 하기 때문에 total로 설정하였다.

15. History_of_shipment_tracking**와 Shipment는 N**(total):1(partial)**의 History_of_Shipment의 관계**

- 운송 중인 소포의 과거 상태(History_of_shipment_tracking)와 shipment는 many to one의 관계를 지닌다. 여러 과거 추적은 하나의 shipment에서 나올 수 있기 때문이다. 그리고 과거 추적에 대한 정보가 존재한다면 반드시 shipment와 관계가 형성되어야 하기 때문에 total으로 설정하였다.

<3> Relational Schema Diagram

위의 e-r diagram을 고려하여 Relational schema diagram을 구성하였다.

먼저 table은 고객(Customer), 비정기적 고객(Infrequent customer), 정기적 고객(Regular customer), 청구서(Bill), 주문(Order), 창고(Warehouse), 소포(Package), 수신인(Recipient), 출하(Shipment), 현재 배송 추적(Current shipping tracking), 운송(Transportation), 배송 추적 기록(History of shipment tracking), 소포 창고(package Warehouse)와 출하 창고(shipment Warehouse)로 구성하였다.

Customer:

customer_id(PK)

- data type은 integer 값으로 설정하였고, primary key로 설정하여 Null값을 가질 수 없도록 하였다.

{phone_number}

- data type은 number로 설정하였고, 한 사람이 여러 휴대전화번호를 가질 수 있으니, multivalued 속성을 주었고, 고객 관리 상 휴대전화 정보는 꼭 필요하다고 생각하여 Null값을 가질 수 없도록 하였다.

address

- data type은 char로 설정하였고, 주소도 package의 픽업 위치가 되거나 recipient의 주소가 될 수 있으므로 정보를 저장해 두는 것이 좋다고 생각하여 Null값을 가질 수 없도록 하였다.

email

- data type은 char로 설정하였고, 이메일은 휴대전화 이외 추가적인 선택 정보라 생각하여 Null값을 가질 수 있도록 하였다.

name

- data type은 char로 설정하였고, 이름은 Null값을 가질 수 없도록 설정하였다. 여기서 이름은 개인 고객의 이름이 될 수 있고, 한 회사의 이름이 될 수도 있을 것이다.

Infrequent customer:

regular_customer_id(PK)

- data type은 integer 값으로 설정하였고, primary key로 설정하여 Null값을 가질 수 없도록 하였다.

{credit_card_number}

- data type은 number로 설정하였고, infrequent customer은 인터넷이나 전화 등을 통하여 신용 카드로 결제하기 때문에 신용카드 번호에 Null값을 가질 수 없도록 하였다. 그리고 한 고객이 여러 신용 카드를 가지고 있을 수 있으니 multivalued 속성으로 표시하였다.

customer_id(FK)

- customer table의 primary key인 customer_id를 참조하고 있다. 관계는 zero or one의 one to one relationship으로 생성하였다. 종종 사용하는 고객에 대한 정보는 customer table에 존재해야 하기 때문에 Null 값은 허용하지 않았다.

Regular customer:

infrequent_customer_id(PK)

- data type은 integer로 설정하였고, primary key로 설정하여 Null값을 가질 수 없도록 하였다.

{account_number}

- data type은 number로 설정하였고, regular customer은 한 달 간의 주문 내역을 계좌 번호로 청구 받기 때문에 Null값을 가질 수 없도록 하였다. 그리고 한 고객이 여러 계좌를 가지고 있을 수 있으니 multivalued 속성으로 표시하였다.

customer_id(FK)

- customer table의 primary key인 customer_id를 참조하고 있다. 관계는 customer와 zero or one의 one to one relationship으로 생성하였다. 정기 고객에 대한 정보는 customer table에 존재해야 하기 때문에 Null 값은 허용하지 않았다.

Bill:

bill_id(PK)

- data type은 integer로 설정하였고, primary key로 설정하여 Null값을 가질 수 없도록 하였다.

total_payment_amount()

- data type은 money로 설정하였고, 한달 간의 모든 주문에 대한 금액이 측정될 것이므로 Null값을 가질 수 없도록 하였다. 만약 한달 동안 주문을 하지 않은 경우에는 0원이라는 금액이 책정될 것이다.

payment_date

- data type은 date로 설정하였고, 청구 금액에 대해 이 날에 지불을 하였다 라는 정보를 확인할 수 있으므로 아직 지불을 하지 않은 경우에는 지불일이 없을 수 있으므로 Null값을 허용하였다. 그리고 고객의 order에 대한 금액에서 총 금액을 구할 수 있으므로 derived 속성으로 설정하였다.

payment_status

- data type은 char로 설정하였고, 고객의 유형에 따라 지불을 이미 완료했을 수도 있고, 아직 지불 이전 등의 상태를 지닐 수 있을 것이다. 따라서 Null값을 가질 수 없도록 하였다.

billing_date

- data type은 date로 설정하였고, 청구일을 의미한다. 예를 들어, 정기 고객이라면 매달 청구일이 똑같을 것이고, 종종 사용하는 고객이라면 청구서를 요청한 날이 될 것이다. 따라서 Null값을 가질 수 없도록 하였다.

customer_id(FK)

- customer table의 primary key인 customer_id를 참조하고 있다. 하나의 customer에 여러 bill을 가질 수 있는 zero, one or more의 one to many의 관계이고, bill에 있는 customer_id는 Null값을 가질 수 없도록 하였다.

Order:

order_id(PK)

- data type은 integer로 설정하였고, primary key로 설정하여 Null값을 가질 수 없도록 하였다.

order_price

- datya type은 money로 설정하였고, 고객이 주문을 할 때, 한 주문에 대한 금액이 바로 책정될 것이므로 Null값을 가질 수 없도록 하였다.

payment_status

- data type은 char로 설정하였고, 고객의 유형에 따라 한 주문에 대한 지불을 이미 완료했을 수도 있고, 아직 지불 이전 등의 상태를 지닐 수 있을 것이다. 따라서 Null값을 가질 수 없도록 하였다.

payment_date

- data type은 date로 설정하였고, 하나의 주문 금액에 대해 이 날에 지불을 하였다 라는 정보를 확인할 수 있으므로 아직 지불을 하지 않은 경우에는 지불일이 없을 수 있으므로 Null값을 허용하였다.

customer_id(FK)

- customer table의 primary key인 customer_id를 참조하고 있다. 하나의 customer가 여러 주문을 할 수 있으므로 zero, one or more의 one to many 관계로 설정하였고, 주문이 들어간 고객에 대한 정보는

반드시 customer table에 존재해야 하므로 Null값을 허용하지 않았다.

package_id(FK)

- package table의 primary key인 package_id를 참조하고 있다. 하나의 주문 당 하나의 package가 발생한다고 가정하였기 때문에 cardinality value가 1인 one to one의 관계를 설정하였다. 또한 order table에 존재하는 어떤 package를 주문하였는지에 관한 정보는 package table에 반드시 존재해야 하므로 Null값을 허용하지 않았다.

Package:

package_id(PK)

- data type은 integer로 설정하였고, primary key로 설정하여 Null값을 가질 수 없도록 하였다.

content_value

- data type은 money로 설정하였고, 세관 신고가 필요하지 않더라도 이 package가 1~5만원 혹은 5~10만원의 가격대이다 라는 정보를 저장하고 있는 것이 운송회사의 측면에서 좋다고 생각하여 Null값을 허용하지 않았다.

content_description

- 어떤 package인지 설명할 수 있는 속성으로, data type은 char로 설정하였고, 세관 신고가 필요할 경우 혹은 특수한 경우 작성할 수 있도록 Null값을 허용하였다.

timeliness_of_delivery

- Data type은 datetime day to day로 설정하였고, Null 값은 허용하지 않았다. Delivery의 우선순위를 날짜로 제한하였고, 약속된 배송시간(day)을 통해 우선순위를 설정할 수 있기에 Null값을 허용하지 않았다.

package_weight

- Data type은 float으로 설정하였고, package의 무게도 주문 금액이나 운송수단 등에 영향을 미치는 요소로, Null값을 허용하지 않았다.

package_type

- Data type은 char로 설정하였고, package의 type도 주문 금액이나 운송수단 등에 영향을 미치는 요소이고, service type으로 package를 구분할 수 있도록 Null값을 허용하지 않았다.

package_origin

- Data type은 char로 설정하였고, package의 픽업 위치로 반드시 존재해야 하는 정보이기 때문에 Null값을 허용하지 않았다.

package_destination

- Data type은 char로 설정하였고, package의 배달 위치로 반드시 존재해야 하는 정보이기 때문에 Null 값을 허용하지 않았다.

Recipient_id(FK)

- Recipient table의 primary key인 recipient_id를 참조하고 있다. 한 명의 recipient는 여러 package를 받을 수 있으므로 one or more의 one to many로 관계로 설정하였다. Recipient가 존재하면 그에 해당하는 package가 존재해야 하고, package에 있는 recipient_id는 Null값이 될 수 없기 때문에 Null값을 허용하지 않았다.

Shipment_id(FK)

- Shipment table의 primary key인 shipment_id를 참조하고 있다. 하나의 shipment에는 여러 package가 포함될 수 있으므로 one to many의 관계로 설정하였다. 그리고 아직 픽업을 하지 않은 경우 shipment에 포함되지 않을 수 있으므로 Null값을 허용하였다.

Recipient:

recipient_id(PK)

- data type은 integer로 설정하였고, primary key로 설정하여 Null값을 가질 수 없도록 하였다.

recipient_address

- data type은 char로 설정하였고, recipient의 주소가 package_destination일 경우가 많겠지만, package의 수령지가 아닐 수도 있고 이사를 갈 수 있는 여러 상황을 고려하여 recipient table에 따로 속성으로 저장하였고, Null값은 가질 수 있도록 하였다. Recipient에 관한 정보가 자세하게 없을 경우 Null값을 줄 수밖에 없을 것이다.

delivery_completion_time

- data type은 datetime day to minute으로 설정하였고, 아직 배송이 완료되지 않은 경우를 고려해 Null 값을 허용하였다.

delivery_departure_time

- data type은 datetime day to minute으로 설정하였고, 아직 배송이 출발되지 않은 경우를 고려해 Null 값을 허용하였다.

promised_delivery_time

- data type은 datetime day to minute으로 설정하였고, 약속된 배송시간은 Null값을 허용하지 않았다.

delivery_status

- data type은 char로 설정하였고, 배송 전/중/후에 관한 정보를 알 수 있도록 Null값을 허용하지 않았다.

customer_id(FK)

- customer table의 primary key인 customer_id를 참조하고 있고, recipient가 customer일 수도 있지만, 수령만 하고, 주문을 한 고객이 아닐 수도 있는 경우를 고려하여 Null값을 허용하지 않았다. 그리고 한 명의 고객은 한 명의 recipient를 의미할 수 있으므로 zero to one의 one to one 관계를 설정하였다.

Shipment:

shipment_id(PK)

- data type은 integer로 설정하였고, primary key로 설정하여 Null값을 가질 수 없도록 하였다.

shipment_origin

- data type은 char로 설정하였고, 하나의 shipment가 어디에서 출발하였는지(예를 들자면 창고, 혹은 다량의 주문이 들어온 한 회사 등)를 알 수 있도록 Null값을 허용하지 않았다.

shipment_destination

- data type은 char로 설정하였고 하나의 shipment가 어디로 향하는지(예를 들자면 창고, 혹은 회사, 특정 구역 등)를 알 수 있도록 Null값을 허용하지 않았다.

shipment_date

- data type은 date로 설정하였고, 언제 배송이 이루어졌는지 알 수 있는 속성이다. 다만 여러 package들로 shipment가 구성은 되었으나 아직 출발하지 않은 경우를 고려해 Null값을 허용하였다.

Warehouse:

warehouse_id(PK)

- data type은 integer로 설정하였고, primary key로 설정하여 Null값을 가질 수 없도록 하였다.

Location

- data type은 char로 설정하였고, 창고에 관한 위치는 Null값을 가질 수 없도록 하였다.

Package_Warehouse

Package_id(PK, FK)

- primary key이며, package table의 primary key인 package_id를 참조하고 있다. 따라서 Null값을 허용하지 않았다.

Warehouse_id(PK, FK)

- primary key이며, warehouse table의 primary key인 warehouse_id를 참조하고 있다. 따라서 Null값을 허용하지 않았다.
- Package_warehouse table의 경우 원래 many to many 관계였던 package와 warehouse의 관계를 table을 생성하여 many to many 관계를 바꿔주었다.

Shipment_Warehouse

shipment_id(PK, FK)

- primary key이며, shipment table의 primary key인 shipment_id를 참조하고 있다. 따라서 Null값을 허용하지 않았다.

Warehouse_id(PK, FK)

- primary key이며, warehouse table의 primary key인 warehouse_id를 참조하고 있다. 따라서 Null값을 허용하지 않았다.
- shipment_warehouse table의 경우 원래 many to many 관계였던 shipment와 warehouse의 관계를 table을 생성하여 many to many 관계를 바꿔주었다.

Current_Shipping_Tracking:

tracking_id(PK)

- data type은 integer로 설정하였고, primary key로 설정하여 Null값을 가질 수 없도록 하였다.

current_location

- data type은 char로 설정하였고, 현재 운송 중인 shipment에 대하여 위치를 알 수 있도록 Null값을 허용하지 않았다.

current_location_time

- data type은 datetime day to minute으로 설정하였고, 현재 운송 중인 시간 정보를 알 수 있도록 Null값을 허용하지 않았다.

currently_headed_location

- data type은 char로 설정하였고, 현재 향하고 있는 위치를 알 수 있도록 Null값을 허용하지 않았다.

delivery_status

- data type은 char로 설정하였고, 현재 운송 중인 정보에 대해서만 이 table에 저장 할 것이므로 Null값을 허용하지 않았고, 운송 중인 상태가 운송 완료로 바뀐다면 그러한 tuple은 이 table에서 삭제될 것이다.

shipment_id(FK)

- shipment table의 primary key인 shipment_id를 참조하고 있고, 하나의 shipment 당 운송 중이라면 하나의 현재 운송 중인 shipment에 대한 추적이 발생할 수 있으므로 zero or one의 one to one의 관계로 설정하였다. 그리고 현재 운송중인 shipment를 추적하고 있다면, shipment_id가 이 table에 반드시 존재해야 하므로 Null값을 허용하지 않았다.

transportation_id(FK)

- transportation table의 primary key인 transportation_id를 참조하고 있고, 하나의 transporation 당 여러 shipment의 운송이 발생할 수 있으므로, one or more의 one to many 관계를 설정하였다. 현재 운송중에 있다면 반드시 어떠한 운송 수단을 이용해 운송 중인 것이므로, Null값을 허용하지 않았다.

History_of_Shipment_Tracking:

tracking_history_id(PK)

- data type은 integer로 설정하였고, primary key로 설정하여 Null값을 가질 수 없도록 하였다.

Location

- data type은 char로 설정하였고, 과거 운송할 당시의 위치를 저장하고 있으므로 Null값을 허용하지 않았다.

Time

- data type은 datetime day to minute으로 설정하였고, 과거 운송할 당시의 시간을 저장하고 있으므로 Null값을 허용하지 않았다.

shipment_id(FK)

- shipment table의 primary key인 shipment_id를 참조하고 있고, 한 shipment의 과거 운송에 대한 tracking이므로 Null값을 허용하지 않았다. 그리고 하나의 shipment에 여러 개의 운송 이력이 남아있을 수 있으므로, zero, one or more의 one to many 관계를 설정하였다. 아직 shipment가 운송되지 않았을 경우나 막 배송을 시작한 경우에는 과거 이력이 없을 것을 고려하였다.

tracking_id(FK)

- current_shipping_tracking table의 primary key인 tracking_id를 참조하고 있고, 하나의 현재 추적이 시간이 바뀌면서 여러 과거 추적 정보로 저장될 수 있으므로 one to many의 관계로 설정하였다. 그리고 현재 운송중인 shipment에 대해서는 과거 이력이 존재할 수 있으나, 배송이 완료된 shipment에 대해서는 현재 운송 정보인 tracking_id가 없을 것이므로 Null값을 허용하였다.

transportation_id(FK)

- transportation table의 primary key인 transportation_id를 참조하고 있고, 하나의 shipment에 대한 하나의 과거 추적은 하나의 운송 수단과 연결되기 때문에 one to one의 관계를 설정하였다. 그리고 과거

shipment tracking에는 반드시 운송 수단에 대한 정보가 존재해야 하므로 Null값은 허용하지 않았고, Cardinality value를 1으로 설정하였다.

Transportation:

transportation_id(PK)

- data type은 integer로 설정하였고, primary key로 설정하여 Null값을 가질 수 없도록 하였다.

transportation_number

- data type은 integer로 설정하였고, 차 번호, 비행기 고유 번호의 정보를 저장하는 속성이기에 Null값을 허용하지 않았다.

transportation_method_type

- data type은 char로 설정하였고, 운송 수단에 대한 설명으로 Null값을 허용하지 않았다.

driver_id

- data type은 integer로 설정하였고, driver에 대한 id를 저장할 수 있으므로 Null값을 허용하지 않았다.

driver_name

- data type은 char로 설정하였고, driver에 대한 이름을 저장할 수 있으므로 Null값을 허용하지 않았다.

{driver_phone_number}

- data type은 number로 저장하였고, 한 명의 driver가 여러 개의 전화 번호를 가질 수 있으므로 multivalued 속성으로 표시하였다. 그리고 회사와 driver의 연락 수단이 전화 번호라고 생각하여 Null값을 허용하지 않았다.