데이터베이스시스템

CSE4110-02

〈Project #1〉

E-R design and Relational Schema design

서강대학교 경제학과

20160563 송진아

목 차

1 프로젝트 개요

이 프로젝트는 주어진 조건에 알맞은 Package Delivery System 에 관한 관계형 데이터베이스를 디자인하는 것을 목표로 한다. 프로젝트 명세서로부터 E-R 을 디자인하고 이로부터 Relational Schema 를 디자인한다. E-R diagram 의 notation 은 교재 『Database System Concepts 7th edition, McGraw-Hill Book Company』 6 장에서 소개된 방식을 따른다. Relational Schema 는 'ERwin' 프로그램에서 작성하며, notation 은 IE 방식을 따른다.

현실에서의 데이터베이스는 많은 정보의 종류 및 양에도 불구하고 예외가 발생하지 않도록 엄밀하게 디자인 해야한다. 예컨대 고객은 여러 개의 주소를 등록할 수도 있고 이 중에서 대표 주소를 지정할 수도 있는 식이다. 그러나 본 프로젝트에서는 주어진 조건을 만족하는 한에서 불필요하게 중복되는 요소를 지양하며 가장 단순한 모델을 생성하고자 하였다. 이는 '2 Entity 와 Relationship 의 정의'에서 보다 자세히 다룰 것이다.

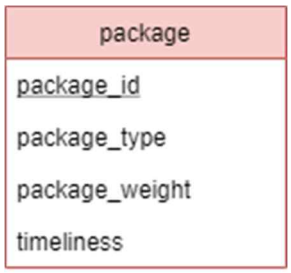
2 Entity 와 Relationship 의 정의

프로젝트 명세서의 "Application Description"에서 주어진 조건을 순서대로 조건 1~6 이라 하자. 조건의 순서에 따라, 관련된 Entity 와 Relationship 을 서술한다.

- 조건 1 : 'package' entity set

  service 의 종류는 type of package, weight of the package, timeliness of delivery 에 의거해 나뉜다. 즉, 하나의 package 는 이 세 가지 종류의 속성을 가지며 회사는 어떤 서비스를 제공했는지 기록하기 위하여 이를 저장해야 한다. 이에 package entity set 을 정의하였다. 또한 package 를 특정하기 위한 'package_id' 를

primary key 로 부여하였다. 예를 들어, package 102 는 작은 박스에 담겨있고, 200g 이며, 익일 배송을 원칙으로 한다.

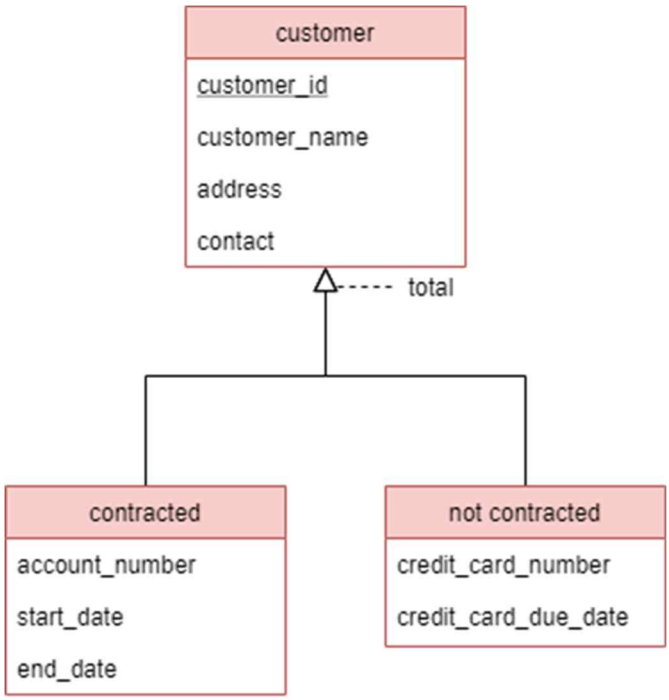


- 조건 2 : 'customer' entity set 과 specialization

배송 서비스의 customer 는 발송인 및 수신인으로 정의할 수 있다. 추가적으로 본 프로젝트에서는 착불의 경우는 존재하지 않는다고 가정하자. 그렇게 하면 배송 서비스의 구매자는 항상 발송인이 된다. 발송인은 명세서에 따라 다음의 두 종류로 나뉜다. 일부는 회사와 계약을 맺어 발송 시점에 배송비를 지불하지 않고 매달 지정일에 청구서를 받아 한 번에 지불한다. (추후 청구서에 관한 서술에서 "past month", "owed" 등의 표현이 있으므로 이 경우는 후불에 해당한다.) 반면 나머지는 신용카드로 지불한다. 이는 발송인이 배송 시작 이전에 선불로 결제하는 경우이다. (신용카드 대금의 입금 시점은 결제 시점 이후가 되지만, 회계상으로도 이 기간 동안의 대금의 공백은 외상 매출금이 아닌 미수금으로 따로 처리하므로 선불로 보아도 무방하다.) 한편 반품의 경우는 최초 물건의 수신인이 발신인이 되어 새로운 배송 서비스를 구매하는 것이다. 따라서 특수하게 고려할 필요가 없다. 착불은 존재하지 않는 것으로 가정하였으므로, 반품 시에는 반품하는 고객이 그의 계약 여부에 따라 배송 서비스 비용을 지불한다.

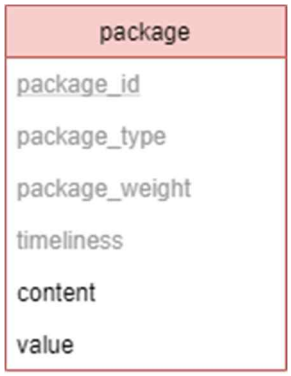
앞서 서술한 사항들에 따라 계약을 맺은 고객과 그렇지 않은 고객은 customer entity set 의 disjoint total specialization 이 된다. customer entity 에는 primary key 로 'customer_id'를 부여하였다. 또한 배송 시에 필요한 이름(customer_name), 주소(address), 연락처(contact) 정보를 속성으로 하였다. 현실에서 주소와 연락처는 여러 개가 될 수도 있지만, 배송에 필요한 가장 최근의 정보 하나만을 저장하는

것으로 가정하였다. contracted entity set 에는 계좌번호(account_number), 계약 시작일(start_date), 계약 만료일(end_date)을 속성으로 추가하였다. 계약 만료 시점이 되면 해당 고객은 not contracted 로 변환된다. 간소화를 위해, 계좌주는 customer_name 과 동일하다고 간주하였다. not contracted entity set 에는 신용카드 번호(credit_card_number)와 신용카드 만료일(credit_card_due_date)을 속성으로 추가하였다. 실제 결제 시에는 이들 외에도 비밀번호 앞자리와 CVC 번호 등을 입력해야 하겠지만, 이들은 보안 상 데이터베이스에 저장하면 안 되는 정보들이다.



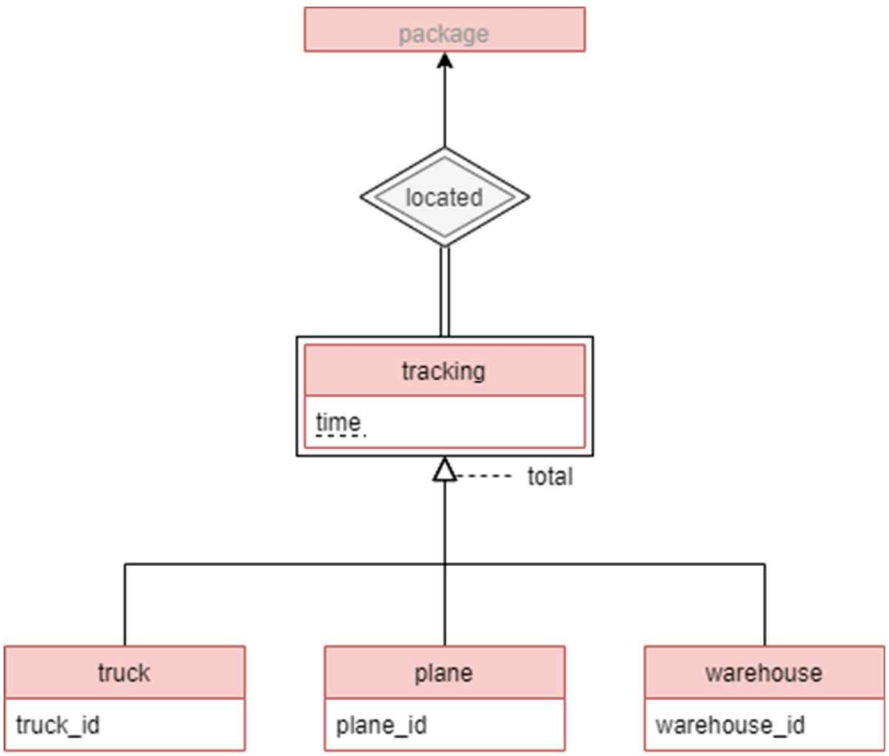
- 조건 3 : 'package' entity set

회사는 일반적으로 배송하는 물건과 물품 가액을 모르지만, 특수한 경우에 한해서는 이 정보를 고객으로부터 받아 저장한다. 이 경우 사측이 먼저 물건의 종류를 판별할 수는 없으므로 고객 측에서 요청하여 물건의 종류 및 가액을 입력할 것이다. 따라서 정보의 입력은 외부에서 주어지며 입력이 없는 경우엔 null 이 될 것이다. 택배 내용물과 물품 가액은 package 를 부가설명하는 것이기 때문에 package 의 속성에 추가하였다.



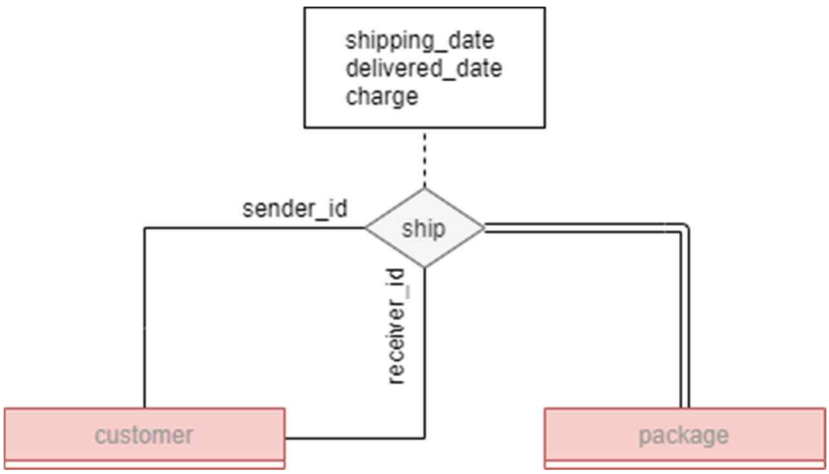
- 조건 4, 5 : ‘tracking’ entity set 과 specialization, ‘located’ relationship set

네 번째와 다섯 번째 조건은 배송 추적에 관한 것이다. 택배가 특정 시점에 어디에 있는지를 나타내며, 택배 한 건 당 여러 개의 추적 기록이 남는다. 트럭, 비행기, 창고는 한 추적 기록의 disjoint total specialization 이 된다. 한 물건이 동시에 여러 장소에 존재할 수는 없기 때문이다. tracking entity set 으로 시점을 표현하고, truck, plane, warehouse entity set 으로 장소를 표현하였다. 그러나 장소와 시간 만으로 하나의 entity 를 특정할 수는 없다. 어떠한 택배인지 대한 정보가 필요하다. 따라서 이들은 weak entity 가 되며 package 와 weak relationship 을 맺는다.



- 기타 : 'ship' relationship set

명세서의 'Project Requirements : 3. Queries' 에 bill 에 대한 설명이 주어진다. 고객, 주소, 지난 한 달의 외상 금액, 서비스의 종류, 각 배송의 가격이 청구서에 포함된다. 먼저 고객과 주소는 customer entity set 에서 정보를 가져올 수 있다. 총 외상 금액은 특정 기간에 해당하는 각 배송 건의 배송비의 합으로 구할 수 있다. 서비스의 종류는 package entity set 에서 정보를 가져올 수 있다. 따라서 entity set 의 추가는 불필요하다고 판단하였다. 대신 ship relationship set 을 추가함으로써 이 정보들을 연결하고 추가적으로 배송을 시작한 날짜(shipping_date), 도착한 날짜(delivered_date), 배송 건에 대한 금액(charge)을 relationship set 의 속성으로 추가하였다. customer entity set 과는 두 번 연결이 되는데, 각각 발송인(sender)과 수신인(receiver)을 의미한다. 비용은 앞서 서술하였듯이 발송인이 지불한다.

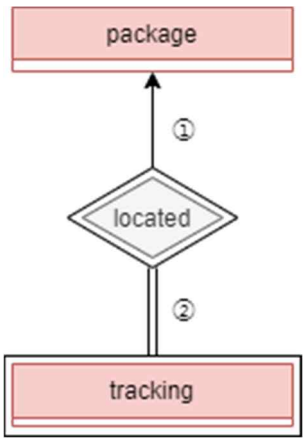


3 Relationship Cardinality 의 정의

앞서 '2 Entity 와 Relationship 의 정의'에서 located 와 ship relationship set 을 생성한 바 있다. '3 Relationship Cardinality 의 정의'에서는 첨부된 이미지로 확인한 relationship cardinality 이 어떻게 정해졌는지 그 이유를 서술한다.

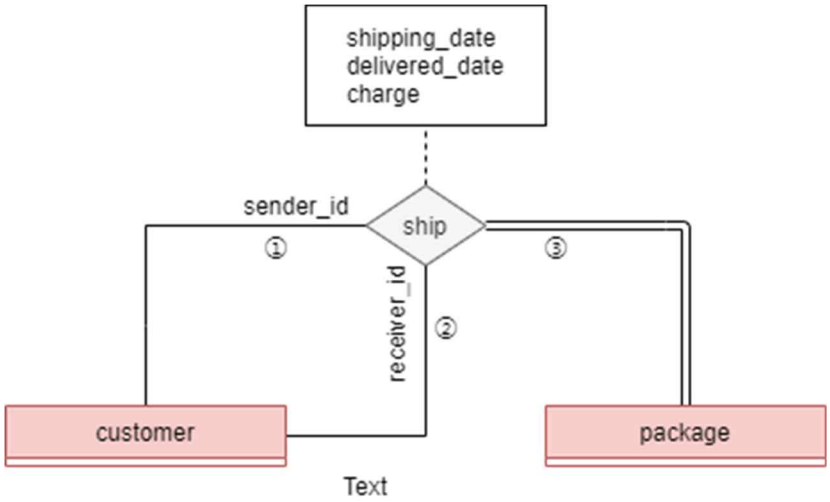
- cardinality of ‘located’ relationship set

  아래 이미지와 같이 one(①) to many(②) 로 정하였다. 하나의 package 에 대하여 tracking(추적 기록)은 여러 개가 존재하고, 하나의 tracking 에 대하여서는 package 를 특정할 수 있어야 하기 때문이다. 한편 모든 tracking 은 package 와 연결되어야 하므로 total(②)이다. 그러나 어떤 package 는 tracking 이 남겨지기 이전이어서 대응되는 tracking 이 존재하지 않을 수도 있기 때문에 partial(①)이다.



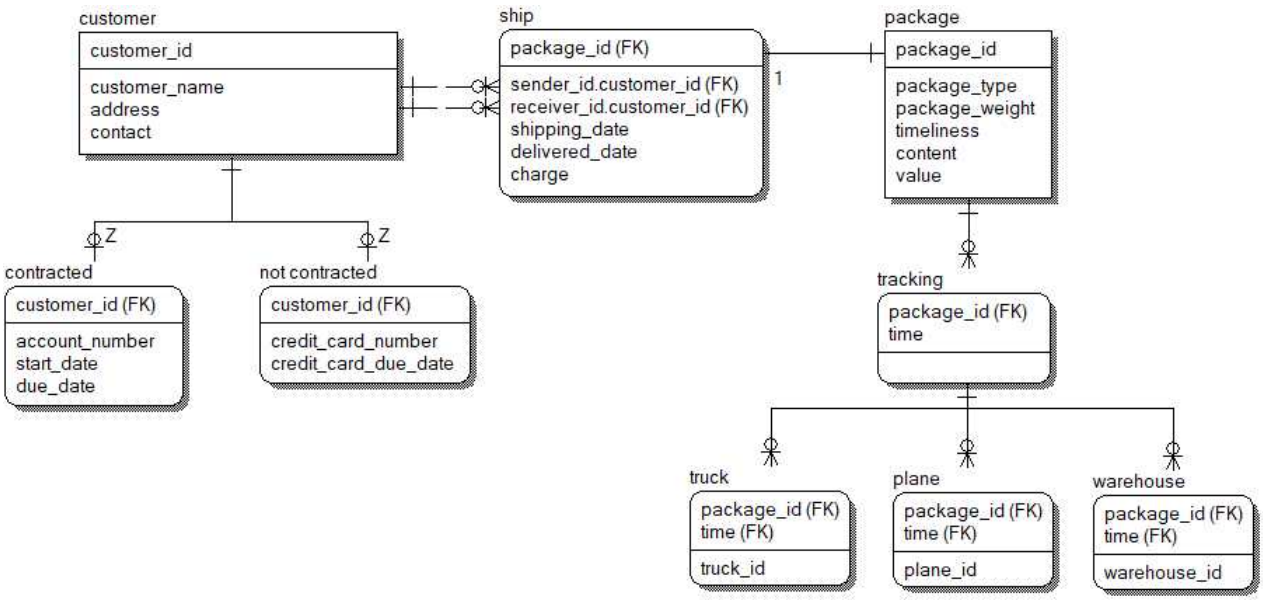
- cardinality of ‘ship’ relationship set

  아래 이미지와 같이 many(①) to many(②) to many(③) ternary relationship 으로 정하였다. 실제로는 하나의 package(③)에 대하여 발송인(①)과 수신인(②)를 특정할 수 있다. 한 물건의 발송인 혹은 수신인이 여러 명이 될 수는 없기 때문이다. 그러나 ternary relationship 에서 one to one to many 와 같은 관계는 해석에 따라 달리 정의할 여지가 있으므로, 엄밀함을 위해 ‘one’은 하나 이하로만 한정하는 규정이 있다. 이에 따라 모두 many 로 지정하였다. 한편 모든 package 에 대해서는 발송인과 수신인이 존재하므로 total(③)이다. 그러나 어떤 customer 는 수신만 해서 발송인으로서는 ‘ship’에 참여하지 않을 수 있기 때문에 partial(①)이 된다.
  마찬가지로 어떤 customer 는 발송만 해서 수신인으로서는 ‘ship’에 참여하지 않을 수 있기 때문에 partial(②)이 된다.



4 Query 의 처리

E-R diagram 을 바탕으로 작성한 schema diagram 에서 명세서에 주어진 queries 를 어떻게 처리할 수 있는지 서술한다. 단, SQL 은 작성하지 않고 간단한 동작 방식만을 설명한다. 각 query 는 명세서에 주어진 순서에 따라 질의 1~5 로 명명한다. 가독성을 위해 relation 이름은 따옴표로, attribute 이름은 기울인 글꼴로 나타낸다. 작성한 schema diagram 은 다음과 같다.



- 질의 1 :

'truck'에서 *truck_id*가 1721 이면서 *time*이 사고 시점인 것들을 모두 찾는다. (*time*은 모든 택배에 대하여 사고 시점의 장소를 특정할 수 있을 정도로 충분히 자주 기록된다고 가정하자. 그렇지 않다면, 가장 마지막에 기록된 위치가 1721 truck 인 것들을 찾아야 한다.) *package_id*로 'ship'을 참조한다. ①*sender_id*로 발송인을 모두 찾고, ②*receiver_id*로 수신인을 모두 찾는다. 또, 'truck'에서 *truck_id*가 1721 이면서 *time*이 사고 시점 이전인 것들을 모두 찾는다. *package_id*로 'ship'을 참조한다. ③*delivered_date*가 가장 큰(최신인) 것을 찾는다.

- 질의 2 :

'ship'에서 *shipping_date*가 지난 1 년 동안인 것을 골라낸다. 이를 *sender_id*에 따라 그룹으로 묶는다. 각 그룹마다 tuple 의 개수를 센다. 그 중 가장 큰 것을 찾는다.

- 질의 3 :

발생주의 회계 원칙상, 현금의 유출입과는 무관하게 거래 시점을 기준으로 장부에 기록한다. 거래 시점이란, 서비스가 이행된 시점을 말한다. 따라서 계약을 맺어 후불하는 고객이든 배송 건 당 선불하는 고객이든 shipping 이 시작된 시점만을 기준으로 비용을 책정하면 된다. 'ship'에서 *shipping_date*가 지난 1 년 동안인 것을 골라낸다. 이를 *sender_id*에 따라 그룹으로 묶는다. 각 그룹마다 *charge*의 합을 구하고 가장 큰 것을 찾는다.

- 질의 4 :

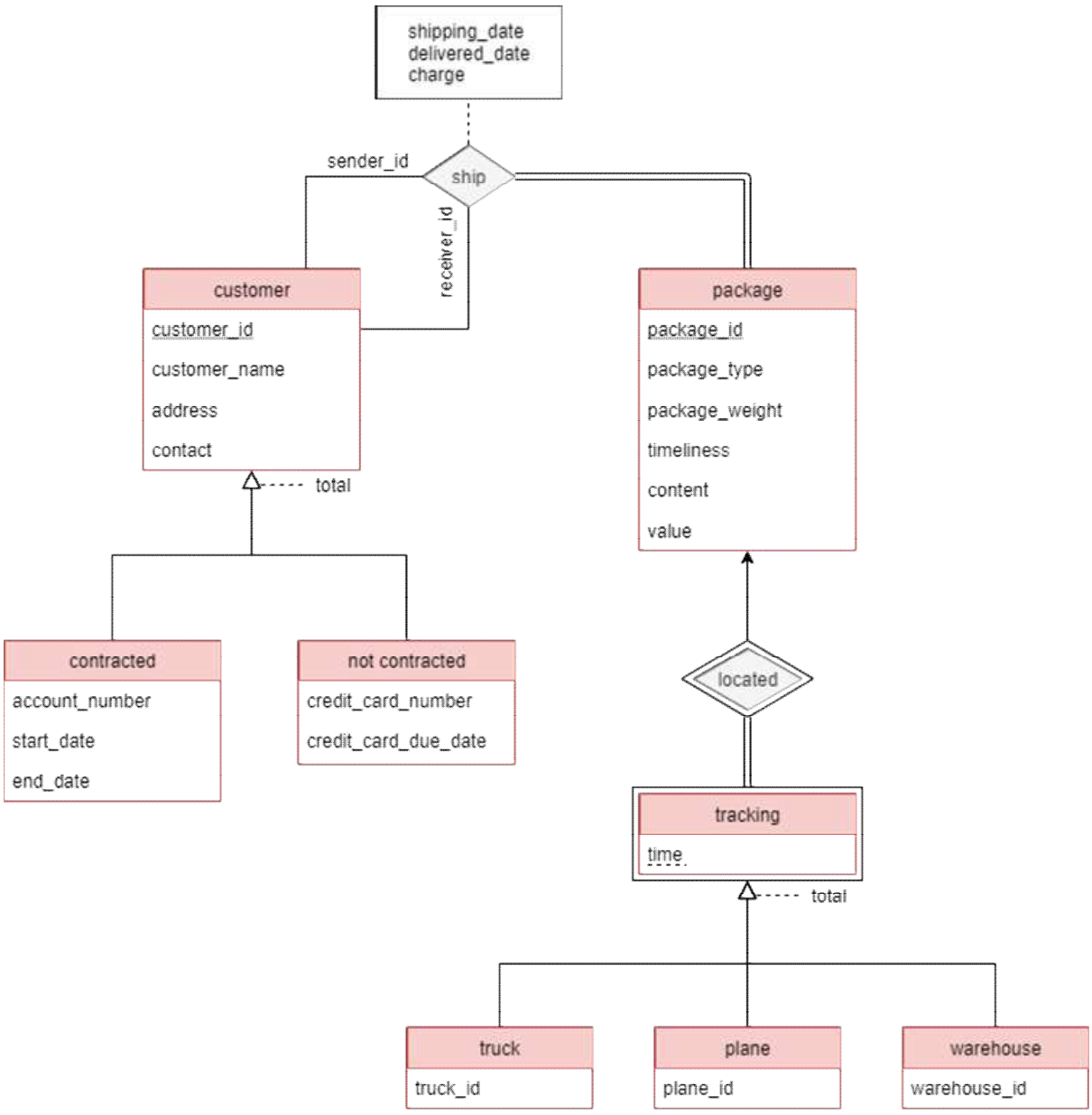
'ship'과 'package'를 *package_id*로 join 한다. *shipping_date*와 *timeliness*의 합보다 *delivered_date*가 큰 것을 찾는다.

- 질의 5 :

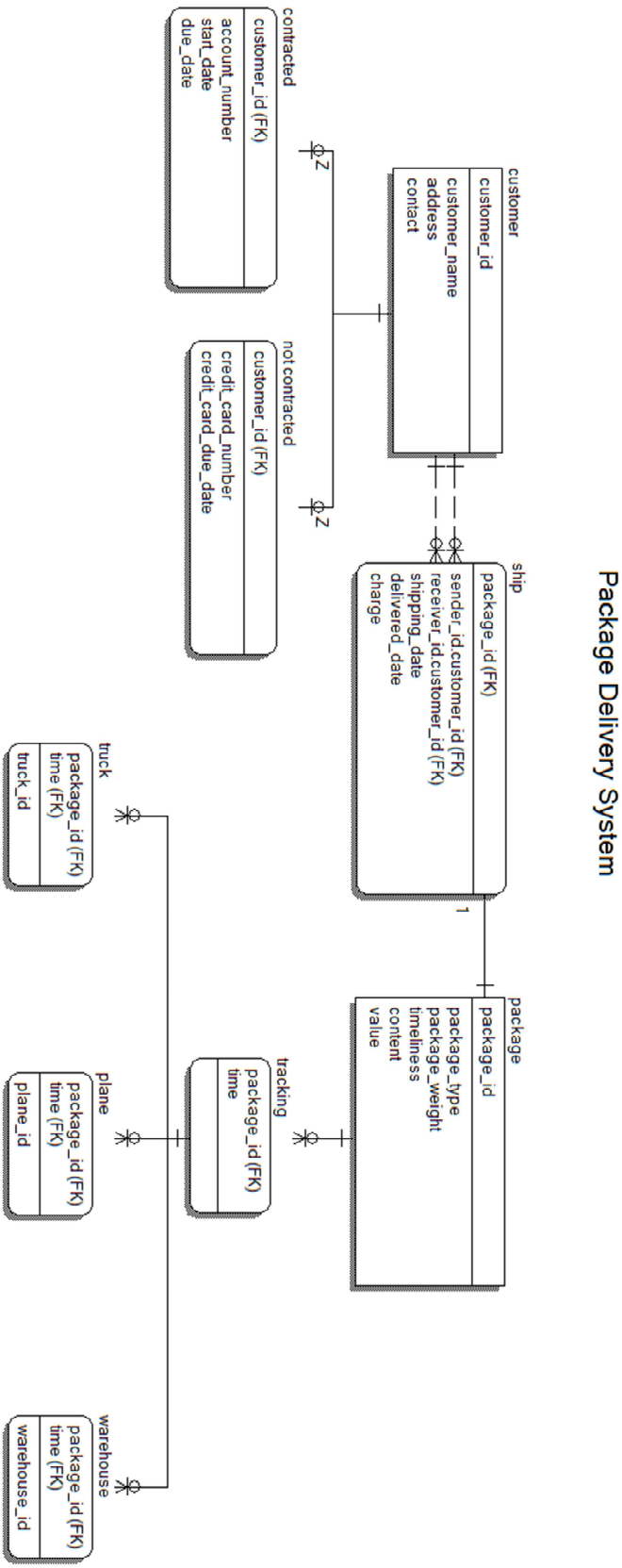
  이전 거래에 대해 청구해야 할 금액이 있는 경우는 계약을 맺어 한 달에 한 번씩 지불하는 고객들에 한정된다. 'ship'에서 *shipping_date*가 지난 달인 것을 골라낸다. *sender_id*에 따라 그룹으로 묶는다. 'contracted'에 *customer_id*로서 존재하는 *sender_id* 그룹 만을 남긴다. 그 다음 ①*sender_id*로 'customer'를 참조하여 *customer_name*, *address*를 구한다. 그룹마다 'ship' *charge* 총액을 구하여 청구한다. 혹은 ②'ship'과 'package'를 *package_id*로 join 한다. 'package' 속성들의 조합에 따라 서비스 종류를 나눈다. 각 종류마다 *charge*의 총액을 구하여 청구한다. ③*package_id*와 *charge* 쌍을 모두 나열한다.

5 부록

- E-R diagram



- schema diagram (logical)



- schema diagram (physical)