데이터베이스시스템
CSE4110-01
[Project1]

서강대학교 컴퓨터공학과

학번 : 20191583

이름 : 김태곤

I. E-R Diagram and Relation Schema Diagram

1. 구현 방법

1) Entity Set

DB 에 필요한 entity Set 을 결정하고 각 entity 에 알맞은 속성을 결정한다.

1-1) Customer

Customer는 고객의 정보가 저장된다. ID, Name, Phone_Number, Address, Contract를 포함하고 있다. ID는 Primary Key이다. ID는 고객을 식별하게 해주는 고유 번호이다. Name과 Phone_Number, Address는 고객의 기본 정보를 담고있다. Phone_Number는 배송 업체에서 고객에게 기타 정보를 제공할 때 연락 수단으로 쓰일 수 있다. Address는 배송지가 잘못된 경우 반송을 위해 필요하다. Contract는 업체와의 계약 관계를 나타낸다. 정기 결제를 한 사람은 bool type True를 나타내고, 이외 일반 고객은 false를 나타낸다.

1-2) Recipient

Recipient 는 배송을 수취인의 정보가 입력된다. ID, Name, Phone_Number, Address 를 포함하고 있다. ID 는 Primary key 로 수취인을 식별하게 해주는 고유 번호이다. Name 은 받는 사람의 이름이 저장된다. Phone_Number 는 수취인이 배송에 대한 정보(도착예정, 도착, 수령방법 등)를 제공 받기 위해 필요하다. Address 는 배송 목적지에 대한 정보를 나타낸다.

1-3) Shipment

Shipment 는 각각의 배송을 나타내는 entity 이다. ID, Items, Order_Time 의 속성을 가지고 있다. ID 는 Primary key 로 배송에 대한 고유 번호를 나타낸다. Items 는 multi-valued attribute 이다. 고객은 배송 업체에 다양한 물건을 한번에 보낼 수 있으므로 Shipment 한번에 Items 를 여러 개 가질 수 있게 구성하였다. Order_Time 은 고객이 배송을 요청한 시간이다. Order_Time 통해 다양한 정보를 얻을 수 있다. (ex 특정 기간을 검색해 총 몇 번 배송 요청이 발생했는지 알 수 있다.)

1-4) Service

Service 는 배송에 대한 부가적인 정보를 나타낸다. Package_Type, Weight, Timeliness 를 포함한다. Service 는 일부의 속성으로는 데이터를 특정할 수 없다. 모든 속성이 있어야 특정한 데이터를 구분할 수 있으므로 세개의 속성 모두 Primary key 이다. Package_Type 은 봉투, 작은 박스, 큰 박스 등 포장 방법이 저장된다. Weight 는 float 형태의 무게가 저장되고, Timeliness 는 지정된 배송 예정 일정이 저장된다.

1-5) Bill

Bill 은 요금 청구에 관한 정보를 나타내는 entity 이다. ID, Pay_Method, Price, Billing_Date, State 의 속성을 가지고 있다. ID 는 Primary key 로 요금 청구서의 고유 번호를 나타낸다. Pay_Method 는 카드결제, 계좌이체, 정기결제 등 결제 방법이 저장된다. Price 는 배송에 의해 발생한 요금이 저장되고, Billing_Date 는 요금 청구가 발생한 날짜(시간)을 저장한다. State 를 통해 고객이 요금을 냈는지 확인할 수 있다.

1-6) Location

Location 은 현재 배송 위치에 대한 정보를 저장한다. Position(Type), Name(ID), Timestamp, State 로 구성되어 있다. Position(Type)은 물건의 위치를 나타낸다. Name(ID)는 구체적인 정보가 저장된다. 예를 들어 패키지가 truck 1721 에 의해 출발한 상태라면 Position(Type)은 Truck 이고, Name(ID)는 1721 이 된다. 용인 Hub 에 도착한 상태라면 Position(Type)은 Hub 가 되고, Name(ID)는 용인이 된다. Timestamp 는 배송 상태의 시간을 나타낸다. 우리가 평소 배송 조회를 할 때 나타나는 형식과 동일하다. State 는 물건의 배송 상태를 나타낸다. Truck 에 의한 물건의 이동은 출발, 도착으로 나타내고, 특정 지역에 도착 후 간선상차, 간선하차 등의 상태로 나타낸다.

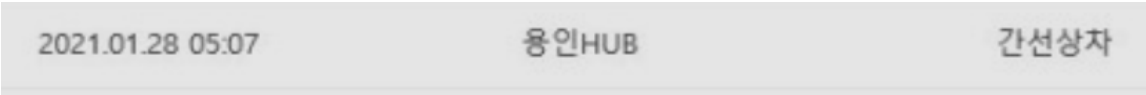

Figure 1 Location 예시

2) Relationship

개체들 간의 적절한 Relation 을 결정한다.

2-1) Cus_Ship

Cus_Ship 은 Customer 와 Shipment 의 관계이다. 한 명의 고객은 여러 개의 배송을 신청할 수 있다. 하나의 배송은 반드시 한 명의 고객에 의해서 신청되고, 모든 배송은 배송을 신청한 고객이 필요하다. 따라서 Shipment 와 Customer 는 Many to One 의 관계이다. Customer 의 Cardinality 는 1..*, Shipment 의 Cardinality 는 1..1 이고, Shipment 는 Total Participation 이다.

2-2) Rec_Ship

Rec_Ship 은 Recipient 와 Shipment 의 관계이다. 하나의 배송에는 한 명의 수취인이 필요하다. 또한 모든 배송에는 수취인이 반드시 있어야 한다. 수취인은 여러 개의 배송을 받을 수 있다. 따라서 Shipment 와 Recipient 는 Many to One 의 관계이다. Recipient 의 Cardinality 는 1..*, Shipment 의 Cardinality 는 1..1 이고, Shipment 는 Total Participation 이다.

2-3) Ship_Ser

Ship_Ser 은 Shipment 와 Service 의 관계이다. 하나의 배송에는 그에 대한 배송정보가 한 개 필요하고, 모든 배송에는 Service 정보가 포함되어 있다. Service 정보는 다른 배송이라도 같은 패키지 타입, 무게, 타임라인이 존재할 수 있으므로, 하나의 Service 정보는 여러 개의 배송과 짝을 이룰 수 있다. 따라서 Shipment 와 Service 는 Many to One 의 관계이다. Shipment 의 Cardinality 는 1..1, Service 의 Cardinality 는 1..*이고, Shipment 는 Total Participation 이다.

2-4) Ship_Bill

Ship_Bill 은 Shipment 와 Bill 의 관계이다. 하나의 배송에는 하나의 Bill 이 필요하다. Bill 이 한 개 있으면 배송 또한 한 개 존재한다. 따라서 Shipment 와 Bill 은 One to One 관계이다. Shipment 의 Cardinality 는 1..1 이고, Bill 의 Cardinality 도 1..1 이다.

2-5) Cus_Bill

Cus_Bill 은 Customer 와 Bill 의 관계이다. Bill 하나에는 청구되는 고객이 반드시 한 명 필요하다. 고객은 요금 청구가 없을 수 있고, 많을 수도 있다. 따라서 Customer 와 Bill 은 One to Many 의 관계이다. Customer 의 Cardinality 는 0..*, Bill 의 Cardinality 는 1..1 이고, Bill 은 Total Participation 이다.

2-6) Ship_Loc

Ship_Loc 는 Shipment 와 Location 의 관계이다. Location 은 배송에 대한 위치 정보를 저장한다. 한개의 배송에 여러 개의 정보가 저장될 수 있다. 배송이 출발하기 전이라면 위치에 대한 정보가 업데이트 되지 않아 없을 수도 있다. 위치에 대한 정보는 반드시 배송이 있어야지 존재한다. 배송이 없다면 Location 은 생성될 수 없으며 종속적이다. 따라서 Shipment 와 Location 은 weak entity 관계이다. Location 의 Cardinality 는 1..1 이고, Shipment 는 0..*이다.

2. 구현 결과

E-R Diagram 과 Relation Schema Diagram 의 구현 결과는 아래 Figure2,3 과 같다.

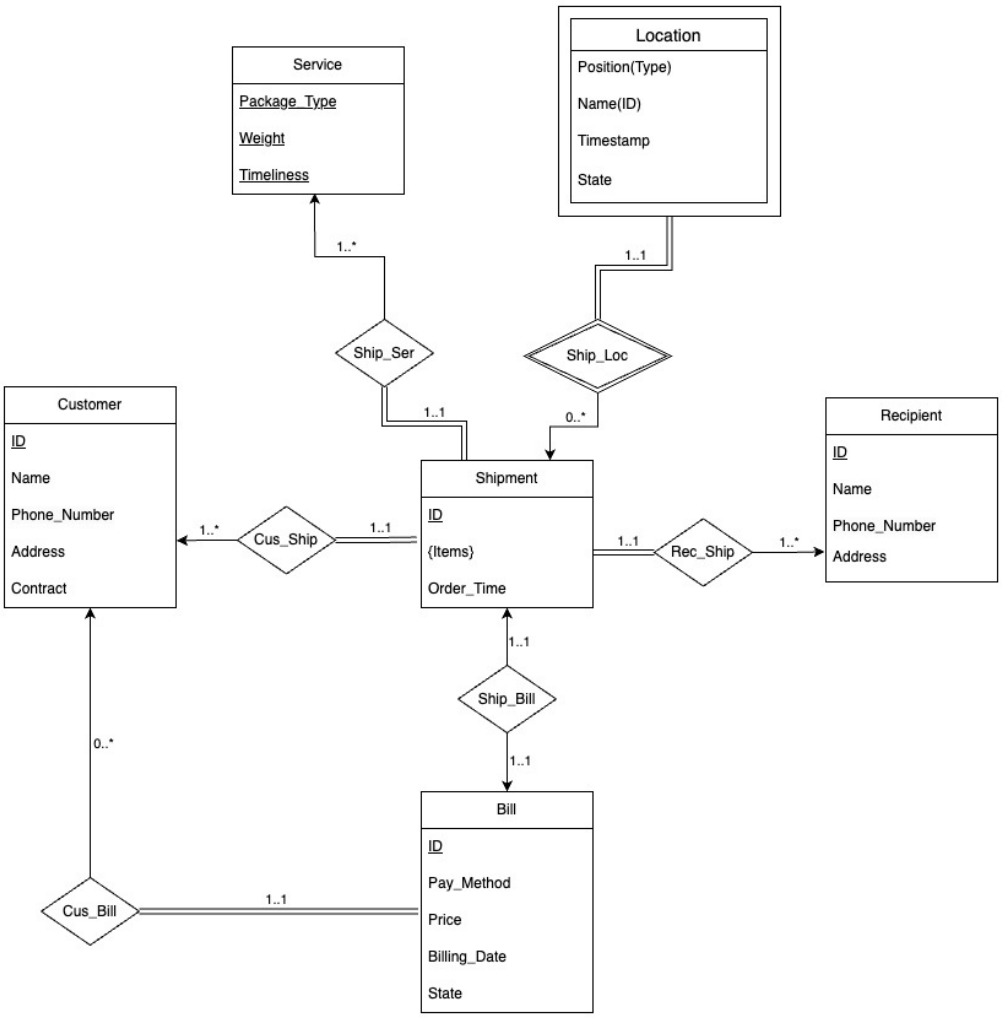

Figure 2 E-R Model

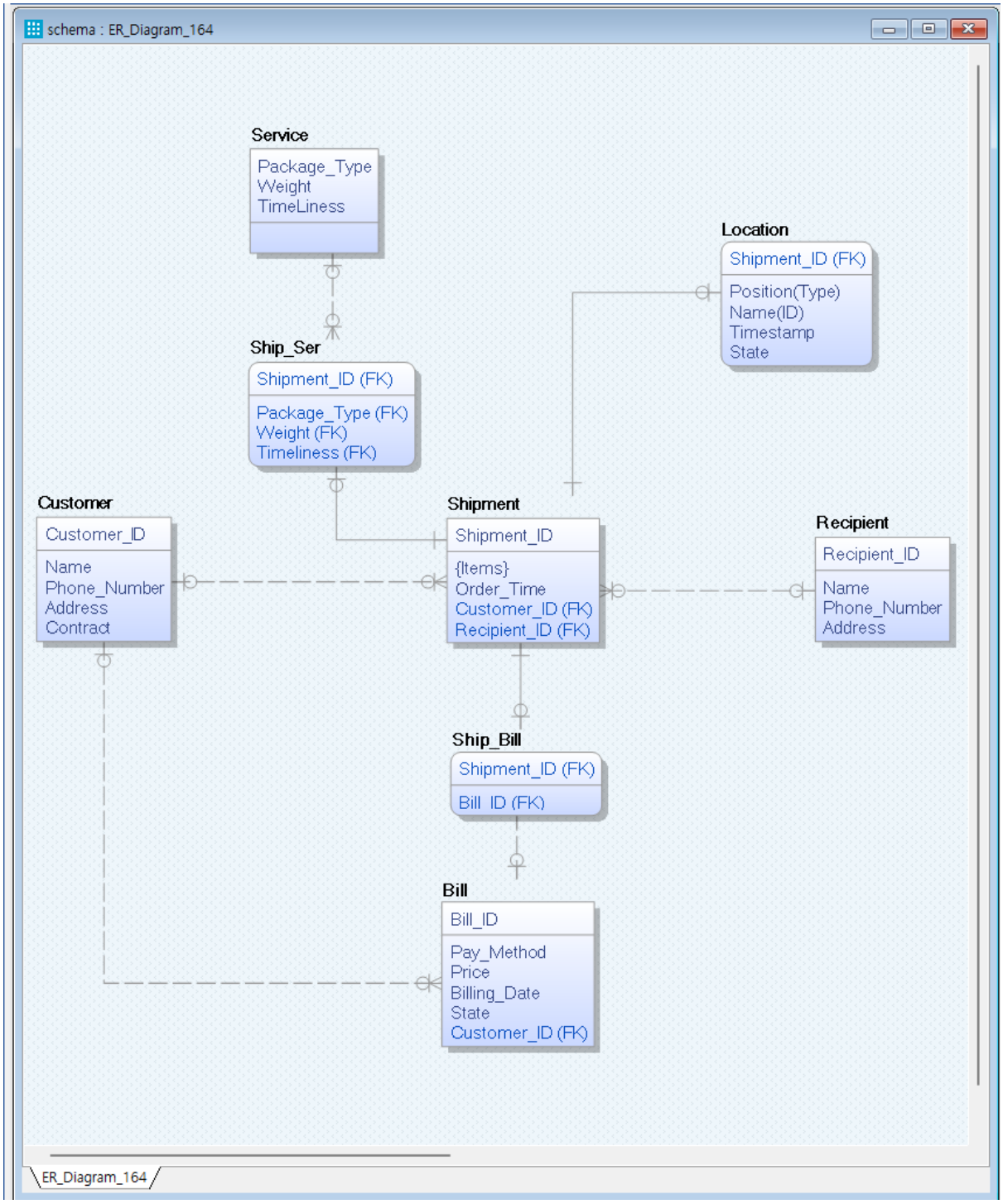

Figure 3 Relational Schema Diagram

II. Queries

과제에 주어진 Queries 에 대해 작성한 DB 에 적용시킨다.

1. 트럭 1721 이 충돌로 인해 파괴되었다고 가정합니다. 충돌 시점에 해당 트럭에 패키지를 보낸 모든 고객을 찾아보세요. 충돌 시점에 해당 트럭에서 패키지를 받은 모든 수취인을 찾아보세요. 충돌 이전에 해당 트럭이 수행한 마지막 성공적인 배송을 찾아보세요.

Sol) Location table에서 (truck, 1721, 출발)이며 충돌시점 T에 가장 가까운 Timestamp를 찾는다.(Position(Type)과 Name(ID)이 같고, 배송 출발 시간이 같으면 같은 트럭이고, 가장 최근에 출발한 트럭이 현재 운행 중 사고가 난 트럭이다.) 앞의 탐색으로 나온 테이블에서 Shipment_ID를 이용해 Customer를 찾는다

-> 사고난 트럭으로 배송 보낸 모든 유저를 찾을 수 있다.

앞에서 찾은 테이블에서 Shipment_ID를 가져오고, 기존 Location 테이블에서 도착 상태인 Shipment_ID를 찾는다. 나온 결과로 Recipient_ID를 찾는다

-> 사고난 트럭이 사고 나기 전까지 배송을 진행해 배송 받은 수취인을 모두 찾을 수 있다.

(truck, 1721, Timestamp<T, 도착) 중 Timestamp가 가장 T에 근접한 것d을 찾는다.

-> 배송 받은 사람들 중 마지막으로 배송 받은 수취인을 찾을 수 있다.

2. 작년 동안 가장 많은 패키지를 발송한 고객을 찾아보세요.

Sol) Shipment에서 Order_Time 기간을 작년 1년으로 잡는다. 나온 결과에 Customer_ID가 같은 것끼리 묶어 수가 가장 많은 것을 찾는다.

-> 작년 한해동안 가장 많이 발송한 고객을 찾을 수 있다.

3. 작년 동안 배송에 가장 많은 비용을 지불한 고객을 찾아보세요.

Sol) Shipment에서 Order_Time 기간을 작년 1년으로 잡는다. 해당 결과의 Customer_ID를 Bill에 검색해 같은 Customer_ID라면 Price를 더해 Total_Price를 생성한다. 이후 Total_Price가 가장 높은 것을 찾아낸다.

-> 작년 한해동안 가장 많은 금액을 지불한 고객을 찾을 수 있다.

4. 약속한 시간 내에 배송되지 않은 패키지를 찾아보세요.

Sol) Position=destination, State=도착인 Location만 뽑는다. 생성된 테이블에서 Shipment_ID를 가져와 Ship_Ser을 통해 Timeliness를 알아내고, Timeliness<Timestamp인 것을 고른다.

-> 약속한 기간 내에 배송되지 않은 패키지들을 모두 찾을 수 있다.

5. 과거 한 달간 각 고객의 청구서를 생성하세요. 다음과 같은 유형의 청구서를 만들 수 있습니다.

간단한 청구서: 고객, 주소 및 미지급 금액.

서비스 유형별로 청구서를 나열한 청구서.

각 개별 발송물과 해당 비용을 나열한 청구서.

Sol) Billing_Date가 한달 이내이고, State가 미납인 것을 뽑아 Customer_ID가 같은것끼리 묶는다(Price는 다 더해 Total_Price를 만든다). Bill의 Customer_ID를 통해 Customer에서 주소와 이름을 join시킨다.

-> 한달간 고객의 미지급 금액 청구서를 만들 수 있다.

-> 위에서 구한 미지급 청구서에서 Pay_Method를 확인해 결제 유형별 청구서를 만들 수 있다.

-> Bill과 Shipment는 1대1 대응이므로 각 발송물과 해당 비용을 나열한 청구서를 만들 수 있다.

이번 과제를 통해 제작한 DB 를 통해 고객들은 다양한 유형의 청구서를 제공받을 수 있다. 또한 배송 추적이 가능하며, 다양한 배송 서비스를 제공받을 수 있다. 주어진 Queries 를 모두 수행 가능하며 이 외에도 다양한 데이터를 효율적으로 다룰 수 있다.