데이터베이스시스템
project1 보고서

학번 : 20191617
전공 : 컴퓨터공학과
이름 : 이규호

- ER diagram의 Entity sets

1) Package

- package_id : package의 식별번호 (not null)
- is_delivered : package의 배달이 완료되었는지 여부 (not null)

2) Shippment

- shippment_id : shippment의 식별번호 (not null)

3) Service

- service_id : package가 가진 serive의 식별번호 (not null)
- weight : package의 무게 (not null)
- type_of_package : package의 type(flat envelope, small box, larger boxes) (not null)
- timeliness : package의 예상 도착 시간 (overnight, second day, or longer) (not null)

4) Customer

- customer_id : customer의 식별번호 (not null)
- name : customer의 이름 (not null)

5) Shipper

- shipper_id : shipper(발송자)의 식별번호 (not null)
- name : shipper의 이름 (not null)
- account_number : shipper의 계좌번호 (not null)

6) Recipient

- recipient_id : recipient(수취인)의 식별번호 (not null)
- name : recipient의 이름 (not null)
- address : recipient의 주소(도착지) (not null)

7) Payment

- payment_id : payment(결제)의 식별번호 (not null)
- is_prepaid : payment의 선불 여부 (not null)
- amount : 지불한 금액 (not null)
- pay_method : payment 방식 (credit card or shipper's account) (not null)
- paid_date : payment가 이뤄진 날짜 (not null)

8) Bill

- bill_id : bill(청구서)의 식별번호 (not null)
- bill_month : bill이 청구된 달 (not null)
- bill_type : bill의 type (not null) (여러가지 type의 bill을 식별하기 위함)

9) Transportation

- transportation_id : transportation(운송수단)의 식별번호 (not null)
- type_of_transport : transportation(운송수단)의 종류 (not null)

- 같은 번호를 가진 두 가지의 transportation을 가지는 경우가 발생하는 것을 고려해 {transportaion_id, type_of_transport}가 primary key이다.

10) Warehouse

- warehouse_id : warhouse(창고)의 식별번호 (not null)

11) Hazardous (weak)

- hazardous_type : hazardous material의 종류 (not null)

12) International

- customs_declaration_id : customs declaration(세관신고)의 식별번호 (not null)
- content : package 안의 content(내용물) (not null)
- value: package 안 content의 value(가치) (not null)

- ER diagram의 Relationship sets

1) pack_ship

- Package와 Shippment 사이의 relationship : many(total)-to-many(total)
- 출발지와 도착지를 공유하는 package의 집합을 Shippment로 정의
- 배송과정에서 하나의 package 또한 여러 shippment가 발생할 수 있기 때문에many-to-many 이다.

2) pack_service

- Package와 Service 사이의 relationship : many(total)-to-one
- weight, type_of_package, timeliness에 따라 service가 정해진다.
- package는 이러한 service를 하나만 가질 수 있기 때문에 many-to-one 이다.

3) cus_pack

- Customer와 Package 사이의 relationship : one-to-many(total)
- customer는 여러 개의 package를 가질 수 있기 때문에 one-to-many 이다.

4) shipper_pack

- Shipper와 Package 사이의 relationship : one-to-many(total)
- shipper는 여러 개의 package를 ship할 수 있기 때문에 one-to-many 이다.

5) recip_pack

- Recipient와 Package 사이의 relationship : one-to-many(total)
- Recipient는 여러 개의 package를 받을 수 있기 때문에 one-to-many 이다.

6) need_declare

- Package 와 International 사이의 relationship : one-to-one
- package는 customs declaration(세관신고)를 최대 하나 받으므로 one-to-one 이다.

7) is_hazardous

- Package 와 Hazardous 사이의 relationship : one-to-many
- Hazardous entity set의 hazardous_type은 discriminator 역할을 하고, package_id를 가져 왔을 때 primary key를 만들 수 있다.
- 따라서 is_hazardous relationship set은 one-to-many 이다.

8) trans_use

- Shippment 와 Transportation 사이의 relationship : many(total)-to-many(total)
- shippment 과정에서 여러 가지 transportation(운송수단)을 가질 수 있고, 하나의 transportation은 여러 shippment를 수행하므로 many-to-many 이다.

9) ware_visit

- Shippment 와 Warehouse 사이의 relationship : many-to-many
- shippment 과정에서 여러 가지 warehouse을 거칠 수 있고, 하나의 warehouse은 여러 shippment가 지나갈 수 있으므로 many-to-many 이다.
- transporation을 통해 한번에 도착지까지 갈 수 있기 때문에 total은 아니다.

10) cus_pay

- Customer 와 Payment 사이의 relationship : one-to-many(total)
- customer는 여러 개의 payment를 할 수 있으므로 one-to-many 이다.

11) shipper_paid

- Shipper 와 Payment 사이의 relationship : one-to-many(total)
- shipper는 여러 개의 payment를 받을 수 있기 때문에 one-to-many이다.

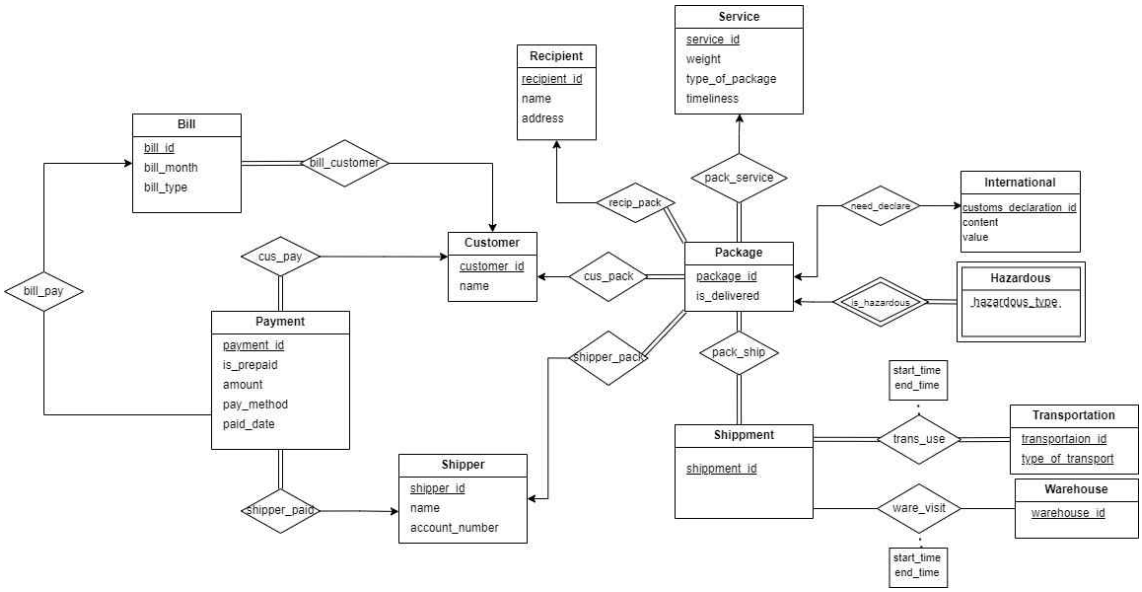
12) bill_pay

- Bill 과 Payment 사이의 relationship : one-to-many
- bill에는 여러 개의 payment가 포함되므로 one-to-many이다.
- 만약 shipper와 계약하지 않고 credit card로 결제하는 payment인 경우에는 bill을 생성하지 않기 때문에 total이 아니다.

13) bill_customer

- Bill 과 Customer 사이의 relationship : many(total)-to-one
- customer는 여러 개의 bill을 받으므로 many-to-one이다.
- 또한 bill은 반드시 해당하는 customer가 존재하므로 total이다.

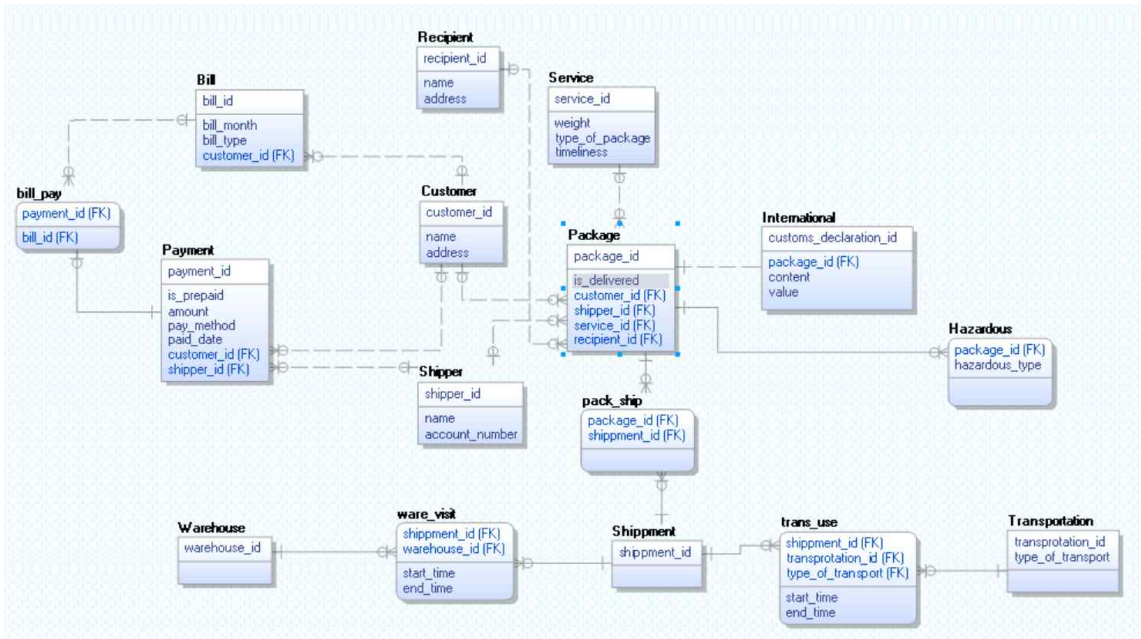
- ER diagram



(tool : draw.io)

- ER diagram에서 Relation Scheme diagram으로 변환

one-to-many(total)의 경우 many쪽의 primary key를 one 쪽의 foreign key로 포함되도록 하였다. 또한 many-to-many의 경우 relationship set을 새로운 table로 만들어 Schema diagram에 표현하였다.(pack_ship, trans_use, ware_visit) 마지막으로 total이 아닌 one-to-many의 경우에는 null을 방지하기 위해 새로운 table을 만들어 표현하였다. (bill_pay)



- Queries

• 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.

- trans_use relation 와 Shippment, Package entity set을 통해 crash가 발생했을 시에 truck에 있던 package들의 customers, recipients, 그리고 tranportation이 가장 최근에 성공한 shippment를 조회할 수 있다.

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

- Payment entity set에서 배송 횟수를 payment 개수를 통해 찾을 수 있다. 또한 paid_date를 통해 기간을 작년으로 한정할 수 있다.

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

- Payment entity set에서 amount와 paid_date를 통해 작년에 사용한 금액의 총합을 구할 수 있고, 그 중 가장 많이 돈을 쓴 사람을 찾을 수 있다.

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

- Service entity set의 timeliness를 통해 예상 배송시간을 알 수 있고, trans_use relation에서 특정 package의 마지막 배송 transportation의 end_time을 통해 실제 도착 시간을 구할 수 있다. 이 둘을 비교하여 약속된 시간 안에 도착하지 못한 package를 찾을 수 있다.

• 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.

- Bill entity set의 bill_type을 통해 bill의 type을 구분할 수 있고, customer과 recipient의 address, amout, type of service, individual shipment 모두 access할 수 있다.