데이터베이스시스템

2분반

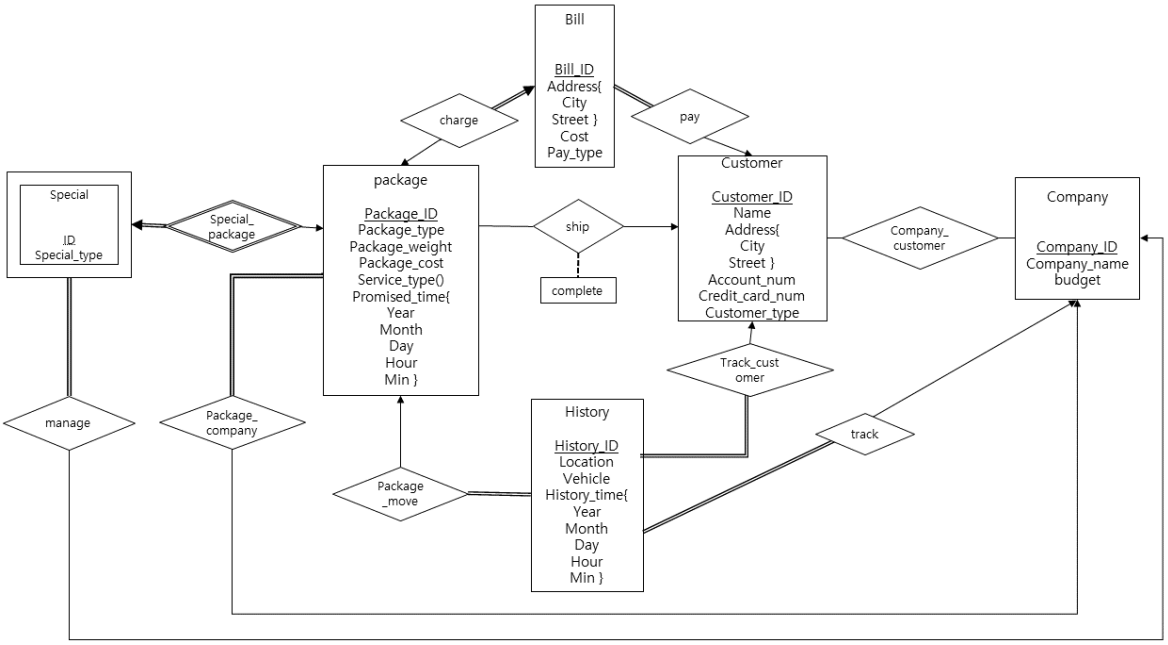
《Project #1》

서강대학교 [컴퓨터공학과]

[20171666]

[이예은]

1. E-R model



a. Entity

Package	Package의 entity. Package의 type, weight, timeline에 따라 service가 달라진다는 명세서를 구현하기 위해 package의 attribute로 package_type, package_weight를 정의했고, 도착 예정 시간인 promised_time을 composite attrebute로 { year, month, day, hour, min }으로 정의했다. 이 세가지 attribute에 따라 service_type이 유도되므로 유도된 attribute로 service_type()로 나타냈다. 또한 다른 attribute로 package의 가격을 나타내는 package_cost와 PK로 package_ID를 넣었다.
Customer	Customer의 entity. PK로 customer_ID를 가진다. 기본적인 customer의 정보로 name, address을 가지며 이 때 address는 composite value이므로 {city, street}로 표현하였다. Customer는 명세서에 따라 monthly pay를 하는 정규고객, infrequent 고객으로 나뉘므로 고객의 type으로 customer_type attribute를 정의하였다. 또한 type에 따라 결제 수단이 다르지만, 정규 고객도 infrequent한 배송도 시킬 수 있다 생각하여 계좌번호와 신용카드 번호를 모두 account_num, credit_card_num attribute로 가지도록 정의했다. 이 때, 이 두 attribute는 계좌번호나 신용카드 번호가 없을 수 있으므로 null이 가능하다.
Company	Company의 entity. PK로 company_ID를 가지며, company의

	기본 attribute로 company_name과 budget을 넣었다.
History	History를 나타내는 entity. PK로 history_ID를 가진다. 명세서에서 which truck or plane 등의 운송 수단과 warehouse를 열람할 수 있게 설계하라 했으므로 attribute로 위치인 location, 운송수단인 vehicle을 가진다. 이 때 이전 시간의 정보까지 열람하기 위해 그 때의 시간을 history_time attribute로 가지게 한다. History_time은 composite attribute로 year, month, day, hour, min을 가진다.
Bill	Bill을 나타내는 entity. PK로 bill_ID를 가지며, 명세서의 Simple bill: customer, address, and amount owed. Bill listing charges by type of service를 구현하기 위해 bill에 attribute로 pay_type과 address를 정의했다. Type of service를 monthly, infrequent, prepaid인 결제 type이라 생각하여 이를 나타내기 위한 pay_type을 정의하였다. 이 때 address는 composite attribute로 city, street를 가지고, 항상 customer의 address에서 결제하지 않고 다른 곳에서도 customer가 충분히 결제할 수 있다 생각하여 address attribute를 relation으로 가져오지 않고 새로 attribute로 정의하였다. 또한 bill의 결제 내용으로 cost를 attribute로 넣었다. 이 때 또한 package_cost에 더해 배송비, 착불비 등이 더해질 수 있으므로 새로 cost attribute를 정의하였다.
special	Company가 package를 신경쓰는 경우에 대해 처리하기 위해 만든 weak entity. Package가 없으면 존재하지 못하므로 package에 종속되었다 말할 수 있고 이로 인해 weak entity로 구현하였다. Special은 PK로 ID를 가지며, 이 package가 hazard나 international package 등 company의 관리가 들어가는 어느 종류의 package인지 구분하기 위해 attribute로 special_type을 넣었다.

b. Relation

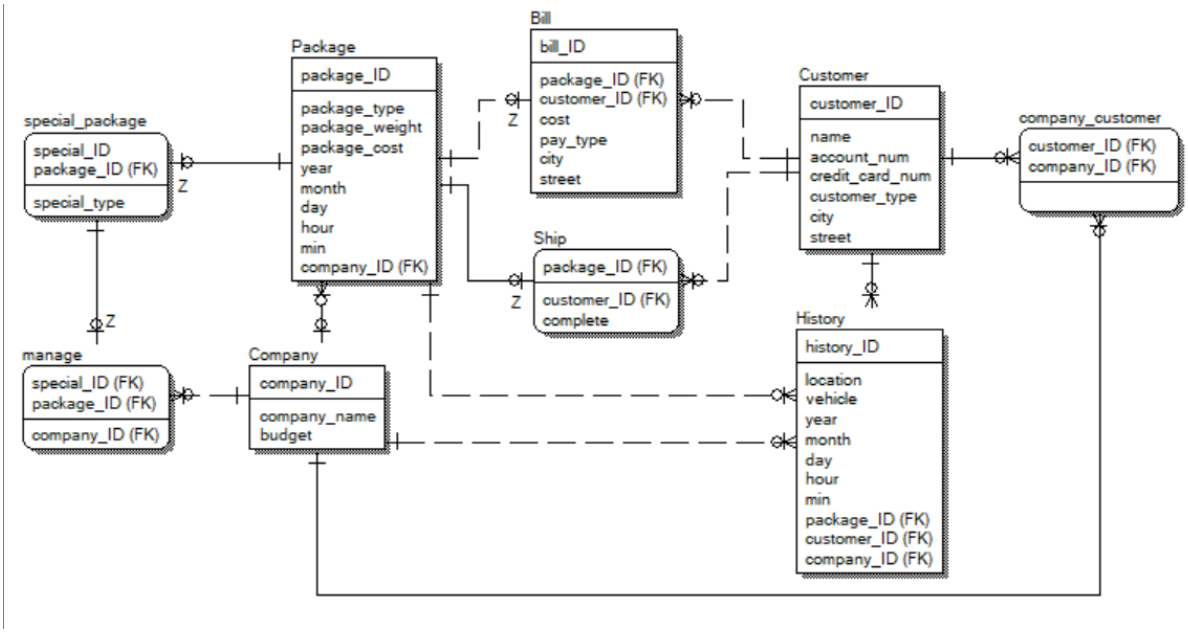
Ship	배송의 relationship으로 package와 customer 사이의 relationship이다. 배송이 완료됐는지 여부를 판단하기 위해 complete attribute를 넣었다. 많은 package는 하명의 customer로 배송될 수 있지만, 하나의 package가 쪼개져 많은 customer로 갈 수는 없다 생각하여 many-to-one cardinality로 정의하였다. 또한 배송이 아직 일

	어나지 않은 package도 있고, 한 번도 배송을 안 시킨 customer도 있다 생각하여 둘 다 partial로 정의했다.
Charge	Package와 bill을 연결하는 relationship. bill에서 어떤 package에 가격이 청구되었는지를 알기 위해 연결하였다. 특히 itemized bill을 출력하기 위해 필요하므로 연결한 relationship이다. 하나의 bill은 반드시 하나의 package를 가지고, 하나의 package 또한 하나의 bill을 가진다 판단하여 one-to-one관계로 정의하였다. 이 때, package에 관한 분할 납부는 없다 생각했고, monthly로 한번에 결제하더라도 각각의 package에 대한 가격 및 지불 내역이 필요하다 생각하여 one-to-one으로 정의하였다. 모든 Bill은 package를 가져야 하지만, 아직 청구되지 않은 package의 경우 bill이 없을 수 있다 판단하여 bill만 total participation으로 정의하였다.
Package_move	Package와 history를 연결하는 relationship. Package의 이동 및 위치, 시간에 대한 정보를 history가 가져야 하므로 연결했다. 많은 package가 하나의 history를 가질 수 없지만, 많은 history가 하나의 package를 가질 수 있다 생각하여 one-to many 관계로 정의하였다. 또한 모든 history는 package를 가지지만, 아직 ship이 일어나지 않은 package의 경우 history가 없을 수 있으므로 history만 total로 정의하였다.
Package_company	Package와 company를 연결하는 relationship. Company가 본인이 가지고 있는 package를 알아야 하기 때문에 연결했다. 많은 package가 하나의 company에 소속될 수는 있지만, 하나의 package가 여러 개의 company에 소속될 수 없다 생각하여 many-to-one의 관계로 정의하였다. 또한 모든 package는 본인을 소유하고 있는 company를 갖지만, 아무 package도 갖지 않는 company도 있을 수 있으므로 package만 total로 정의하였다.
Special_package	Package와 special weak entity를 연결하기 위한 relationship. 하나의 package가 하나의 special한 package가 될 수 있으므로 one-to-one관계로 정의하였다. 또한 special은 반드시 package이어야 하지만, package는 special이 아닌 것도 존재하므로 special만

	total로 정의하였다.
Pay	Customer와 bill을 연결하는 relationship. Customer가 본인이 지불한 내역에 대해 접근할 수 있어야 하므로 연결했다. 많은 bill은 하나의 customer가 지불할 수 있지만, 많은 customer가 하나의 결제내역, bill을 만들 수는 없다 판단하여 customer-bill: one-to-many로 정의하였다. 또한 customer는 bill이 없을 수 있지만, 모든 bill은 지불한 사람인 customer 정보가 있어야 하므로 bill에만 total로 정의하였다.
Track_customer	Customer와 history를 연결하는 relationship. Customer가 본인의 물건이 현재 어디에 있는지, 어느 곳에 있었는지를 확인할 수 있게 하기 위해 history에 접근할 수 있어야 하므로 연결하였다. 한 명의 customer가 많은 history를 가질 수 있지만 하나의 history가 많은 customer를 가질 수 없다 생각해 customer-history: one-to many로 정의하였다. 또한 customer는 history를 가지지 않아도 되지만 history는 반드시 하나의 customer를 가지므로 history를 total로 정의했다.
Company_customer	Company와 customer를 연결하는 relationship. Company가 본인의 고객 정보를 알 수 있어야 하므로 연결하였다. 많은 customer가 많은 company를 가질 수 있고, 반대도 성립하기 때문에 many-to-many relationship으로 정의하였다.
Track	Company와 history를 연결하는 relationship. 명세서에 따라 Company 또한 물건이 어디 있는지, 어디에 있었는지 등을 열람할 수 있어야 하기 때문에 연결했다. 다른 cardinality와 마찬가지로 하나의 history는 하나의 company만 가지고 여러 개의 company를 가질 수 없지만 하나의 company는 여러 개의 history를 가지므로 company-history: one-to-many로 정의하였다. Company는 history가 없을 수 있지만 history는 반드시 company 정보가 들어가므로 history를 total로 정의하였다.
manage	Company가 특정 물건을 관리한다는 명세서 내용에 따라 company와 special을 연결한 relationship. 여러 개의 special package가 하나의 company로 들어갈 수 있

	지만 여러 company가 하나의 special package를 관리하지 않으므로 special-company: many-to-one으로 정의하였다. 또한 모든 special은 company의 관리를 받지만, company는 special한 물품이 없어도 되므로 special에만 total로 정의하였다.

2. Relational Schema Diagram



Package	Package entity의 attribute를 모두 갖고 company_ID를 FK로 가지도록 relation을 합쳐 표현했다.
Customer	ERD Customer entity의 attribute를 모두 갖고 있는 entity로 표현하였다.
Company	ERD Company entity의 attribute를 모두 갖고 있는 entity로 표현하였다.
bill	Bill은 ERD의 bill entity의 attribute를 가지며, package와의 관계는 one-to-one으로 relation이 bill에 나타나도록 합쳤으며 customer와는 many-to-one관계로 many인 bill에 relation을 합쳐 표현하였다. 그래서 customer_ID와 package_ID를 FK로 가진다.
History	ERD의 history entity의 attribute를 가진다. 거기에 customer, package, company와 모두 many-to-one 관계인 것을 이용해 many 측인 history에 relation을 표현했다. 그

	래서 package_ID, customer_ID, company_ID를 FK로 갖는다.
Ship	Package와 customer를 연결하는 relationship을 표현. many쪽인 package의 PK, package_ID를 PK로 가지고, customer의 PK를 FK로 가진다. 또한 배송 여부를 알기 위한 complete attribute를 넣어 표현했다.
Company_customer	Customer-company의 many-to-many 관계를 표현하기 위해 만든 relationship이다. Many-to-many이므로 customer와 company의 PK를 모두 PK로 가진다.
Special_package	ERD의 special weak entity를 나타내기 위한 것이다. Weak entity이므로 package의 PK와 본인의 PK를 모두 PK로 가지며, ERD의 attribute를 가진다.
manage	Company와 special을 연결하기 위해 나타낸 relationship. Special_package의 모든 PK를 PK로 가지고, company의 PK를 FK로 attribute로 가진다.

3. Example Query

명세서의 query 중 하나를 Relational schema로 구현한 예시를 생각해보자.

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

⇨ 우선 package id와 history에 저장된 package id FK를 이용해 package의 history 정보를 뽑아낸다. 이후 package에 저장된 promised time 시간 정보와 history의 history time 정보를 비교해 하나라도 history time이 큰 것이 있다면 promised time이 지나서 이동한 것이 되므로 늦은 배송이 된다. 그러므로 some과 except 등을 이용해 하나라도 history time이 큰 package ID를 제외하고 나머지 package를 뽑아 출력해준다.