Database System

Project1

서강대학교 컴퓨터공학과

20191637

임수민

1. Entity & Attributes 도출

이 프로젝트는 온라인 사이트와 오프라인 체인점을 운영하고 있는 전자제품 판매 회사의 DB 관리자가 되어, 회사 데이터베이스의 개념적 모델링과 논리적 모델링을 수행한다. E-R 모델을 사용하여 개념적 모델링을 수행하고, 생성된 E-R 모델을 바탕으로 Relational Schema Diagram을 생성한다. 먼저 개념적 모델링을 위한 E-R 모델을 작성하기 위해 Entity와 Entity의 Attributes를 도출한다.

1.1 customer

- Entity 도출

전자제품 판매 회사는 고객(customer)을 상대로 제품을 판매한다고 나와있으므로 고객을 관리하기 위해 고객의 정보를 저장하는 customer Entity를 도출했다.

- Attributes 도출

customer Entity는 customer_ID, name, online_id, online_password, online_sign_up_date, email, phone_number, address, birth_date들을 Attribute로 갖는다. 이 때 customer_ID는 primary key로 전자제품 판매회사의 고객에 대해 1 이상의 양의 정수의 값을 가지는 고유한 고객ID를 나타낸다.

name은 고객의 이름을 나타내고, online_id, online_password, online_sign_up_date는 각각 온라인 고객의 온라인 아이디, 온라인 패스워드, 온라인 사이트 가입일자를 나타내고, email, phone_number, address, birth_date는 각각 고객의 이메일, 전화번호, 주소, 생년월일을 나타내는 Attribute다.

1.2 card

- Entity 도출

주어진 고려사항에서 몇몇 고객은 신용카드나 직불카드로 지불한다고 나와있고, 온라인 고객의 카드 정보는 저장되어야 한다고 나와있으므로 card Entity를 도출했다.

- Attributes 도출

card Entity는 card_number, card_company, expiry_date를 Attribute로 가지며, 각각 카드번호, 카드 회사, 유효기간을 나타내는 Attribute이다. 이 때 각각의 card를 구분하기 위해 primary key를 card_number와 card_company로 설정하여 두 개의 Attribute를 primary key로 갖는다.

1.3 account

- **Entity 도출**

주어진 고려사항에서 몇몇 고객은 구매 대금을 계좌 번호로 청구하며 달마다 청구한다고 나와 있으므로, 고객의 계좌 정보를 저장하기 위해 account Entity를 도출했다.

- **Attributes 도출**

account Entity는 account_number, bank_name, account_name, balance를 Attribute로 가지며, 각각 계좌번호, 은행 이름, 계좌명, 잔액을 나타내는 Attribute이다. 이 때 각각의 account를 구분하기 위해 primary key를 account_number와 bank_name으로 설정한다.

1.4 order

- **Entity 도출**

고객이 온라인 사이트나 오프라인 상점에서 제품을 구매했을 때 생기는 주문/구매 정보를 저장하기 위해 order Entity를 도출했다.

- **Attributes 도출**

order Entity는 order_date, order_ID, total_price를 Attribute로 가진다. order_date는 주문일자(구매일자)를 나타내며, order_ID는 주문 정보에 대해 1 이상의 양의 정수의 값을 가지는 고유한 주문번호를 나타낸다. 이 때 각각의 order를 구분하기 위해 primary key를 order_date와 order_ID로 설정한다.

total_price는 고객이 주문한(구매한) 모든 제품의 총 금액을 나타내는 Attribute다. 클라이언트가 요구하는 쿼리 리스트 중 "작년에 가장 많이 산 고객을 찾는" 쿼리가 있으므로 고객이 주문한(구매한) 모든 제품의 총 금액을 나타내는 total_price 속성을 추가하여 이후 customer_ID와 total_price를 사용하여 위의 쿼리를 구현할 수 있게 한다.

1.5 contract_of_payment

- **Entity 도출**

주어진 고려사항에서 몇몇 고객은 구매 대금을 계좌 번호로 청구하며 달마다 청구한다고 나와 있으므로, 주문/구매 정보와 계좌 정보가 포함된 지불 계약서의 정보를 저장하기 위해 contract_of_payment Entity를 도출했다.

- **Attribute 도출**

contract_of_payment에 추가로 필요한 Attribute는 없다. contract_of_payment Entity의 primary key는 account Entity와 order Entity의 primary key를 foreign key로 받아 이들

의 조합을 primary key로 설정할 것이다.

1.6 bill

- Entity 도출

주어진 고려사항에서 몇몇 고객은 구매 대금을 계좌 번호로 청구하며 달마다 청구한다고 나와 있고, 클라이언트가 요구하는 쿼리 리스트 중 "각각의 고객마다 지난 달에 대한 청구서를 생성하는" 쿼리가 있다. 그러므로 각각의 지불 계약서에 대해 매달 생성되는 청구서에 대한 정보를 저장하기 위해 bill Entity를 도출했다.

- Attributes 도출

bill Entity는 bill_number, bill_date, due_date, monthly_charge, total_charge를 Attribute로 가진다. 이 때 bill_number는 primary key로 매달 생성되는 청구서에 대해 고유한 값을 가지는 청구서 번호를 나타낸다.

bill_date, due_date, monthly_charge, total_charge는 각각 청구일자, 지불 만기일, 월 청구액, 총 청구액을 나타내는 Attribute이다.

1.7 delivery

- Entity 도출

주어진 고려사항에서 online sale은 shipper를 통해서 전달되고 운송 회사에 대한 배송 추적 번호를 저장해야 한다고 나와 있으므로 배송 정보를 저장하기 위해 delivery Entity를 도출했다.

- Attributes 도출

delivery는 tracking_number, sending_date, promised_delivery_date, delivery_date, delivery_address, completed, issue, shipper_name을 Attribute로 갖는다. 이 때 tracking_number는 primary key로 고유한 값을 가지는 배송 추적번호를 나타낸다.

sending_date, promised_delivery_date, delivery_date, delivery_address는 각각 발송일자, 예정 배송일자, 실제 배송일자, 배송주소를 나타낸다. 클라이언트가 요구하는 쿼리 리스트 중 "예정된 시간 내에 배달되지 않은 패키지를 찾는" 쿼리가 있으므로 promised_delivery_date, delivery_date 속성을 추가하여 예정 배송일자, 실제 배송일자 정보를 저장한다.

completed는 아직 배송이 완료되지 않았는지, 배송이 예정 시간에 맞게 완료되었는지, 예정 배송 시간보다 늦게 완료되었는지, 배송이 실패했는지를 나타내는 Attribute이다. 아래의 표는 각각의 값에 따른 completed의 내용이다.

completed	
0	아직 배송이 완료되지 않은 경우
1	배송이 예정 시간에 맞게 완료된 경우
2	배송이 예정 시간보다 늦게 완료된 경우
3	배송이 실패된 경우

클라이언트가 요구하는 쿼리 리스트 중 "예정된 시간 내에 배달되지 않은 패키지를 찾는" 쿼리가 있고, 배송이 사고로 인해 실패되는 경우에 대한 배송 정보를 찾는 쿼리가 있으므로, completed 속성을 추가하여 쿼리를 구현할 수 있게 한다.

issue는 배송이 실패했을 경우 실패 원인 등의 이슈 사항을 나타내는 Attribute이다. 예를 들어 사고로 인해 배송이 실패되면 issue에 'accident'를 저장한다. shipper_name은 배송인을 나타내는 Attribute이다.

1.8 shipping_company

- **Entity 도출**

주어진 고려사항에서 online sale은 shipper를 통해서 전달되고 운송 회사에 대한 배송 추적 번호를 저장해야 한다고 나와 있으므로, 운송 회사에 대한 정보를 저장하기 위해 shipping company Entity를 도출했다.

- **Attributes 도출**

shipping_company는 회사명을 나타내는 company_name을 Attribute로 가진다. 이 때 각각의 shipping_company를 구분하기 위해 company_name을 primary key로 설정한다.

1.9 product_type

- **Entity 도출**

주어진 고려사항에서 제품은 종류(type)에 의해 분류될 수 있다고 나와 있으므로, 제품 종류에 대한 정보를 저장하기 위해 product_type Entity를 도출했다.

- **Attributes 도출**

product_type는 제품 종류를 나타내는 type을 Attribute로 가진다. 이 때 각각의 product_type를 구분하기 위해 type을 primary key로 설정한다.

1.10 product

- **Entity 도출**

전자제품 판매 회사는 제품의 정보를 관리하기 때문에 제품의 정보를 저장하는 product Entity를 도출했다

- **Attributes 도출**

product Entity는 product_ID, product_name, model, price를 Attribute로 갖는다. 이 때 product_ID는 primary key로 전자제품 판매회사의 제품에 대해 1 이상의 양의 정수의 값을 가지는 고유한 제품ID를 나타낸다.

product_name, model, price는 각각 제품명, 모델명, 가격을 나타내는 Attribute이다.

1.11 store

- **Entity 도출**

전자제품 판매 회사는 여러 개의 오프라인 상점(체인점)들을 운영하고 있다고 나와있으므로 상점의 정보를 저장하는 store Entity를 도출했다.

- **Attributes 도출**

store Entity는 store_ID, region, owner, address, phone_number, opening_date를 Attribute로 갖는다. store_ID는 지역별로 각각의 상점에 대해 1 이상의 양의 정수의 값을 가지는 고유한 상점ID를 나타내고, region은 상점의 지역을 나타낸다. 이 때 각각의 store를 구분하기 위해 primary key를 store_ID와 region으로 설정한다.

owner, address, phone_number, opening_date는 각각 상점의 소유주, 주소, 전화번호, 개업일자를 나타내는 Attribute이다.

1.12 warehouse

- **Entity 도출**

전자제품 판매 회사는 웹 사이트 또한 운영하고 있다고 나와있고, 주어진 고려사항에서 온라인 고객에게 배송되는 제품을 보관하는 창고를 관리한다고 나와있으므로, 창고의 정보를 저장하는 warehouse Entity를 도출했다.

- **Attributes 도출**

warehouse Entity는 warehouse_ID, manager, address, phone_number를 Attribute로 갖는다. 이 때 warehouse_ID는 primary key로 온라인 사이트에서 판매되는 제품의 재고를 보관하는 각각의 창고에 대해 1 이상의 양의 정수의 값을 가지는 고유한 창고ID를 나타낸다.

manager, address, phone_number는 각각 창고 관리인의 이름, 창고 주소, 창고 전화

번호를 나타내는 Attribute이다.

1.13 offline_inventory

- Entity 도출

각각의 오프라인 상점에 있는 각각의 제품들의 재고 정보를 저장하기 위한 offline_inventory Entity를 도출한다.

- Attributes 도출

offline_inventory Entity는 각각의 상점에 있는 각각의 제품의 재고 수량을 나타내는 stock을 Attribute로 갖는다. offline_inventory Entity의 primary key는 store Entity와 product Entity의 primary key를 foreign key로 받아 이들의 조합을 primary key로 설정할 것이다.

1.14 online_inventory

- Entity 도출

각 창고에 있는 온라인 사이트에서 판매되는 각 제품들의 재고 정보를 저장하기 위한 online_inventory Entity를 도출한다

- Attributes 도출

online_inventory Entity는 각각의 창고에 있는 각각의 제품의 재고 수량을 나타내는 stock을 Attribute로 갖는다. online_inventory Entity의 primary key는 warehouse Entity와 product Entity의 primary key를 foreign key로 받아 이들의 조합을 primary key로 설정할 것이다.

1.15 sales_data

- Entity 도출

주어진 고려사항에서 매출 정보는 기업 계획에 중요하고, 마케팅 담당자는 기간, 제품, 제품 분류, 시즌, 지역에 따른 매출 정보를 원한다고 나와있으므로, 매출 정보를 관리하기 위해 sales_data Entity를 도출했다. 하나의 sales_data 엔터티는 한 제품이 판매될 때의 총 수량과 총 판매액을 포함하는 매출 정보를 저장한다.

- Attributes 도출

sales_data Entity는 sales_ID, total_amount, total_sales를 Attribute로 갖는다. 이 때 sales_ID는 primary key로 1 이상의 양의 정수의 값을 가지는 고유한 매출 정보ID를

나타내어 각각의 매출 정보를 구분한다.

total_amount, total_sales는 각 제품이 주문/구매될 때의 총 수량과 총 판매액을 나타내는 Attribute이다. 클라이언트가 요구하는 쿼리 리스트 중 "판매액 별 상위 2개 제품을 찾는" 쿼리와 "단위 판매량별 상위 2개 제품을 찾는" 쿼리가 있으므로, sales_data Entity에 total_amount, total_sales 속성을 추가하여 쿼리를 구현할 수 있게 한다.

sales_data Entity의 primary key는 order Entity와 offline_inventory 또는 online_inventory의 primary key를 foreign key로 받아 이들의 조합을 primary key로 설정할 것이다.

1.16 offline_reorder

- Entity 도출

주어진 고려사항에서 재고가 부족하면 제조사에 reorder를 보내고 reorder 리스트를 데이터베이스에 저장한다고 나와있으므로, 오프라인 상점에서 제조사에 보내는 reorder 정보를 저장하는 offline_reorder Entity를 도출했다.

- Attributes 도출

Offline_reorder Entity는 reorder_ID, quantity, reorder_date, promised_arrival_date, arrival_date, completed, issue를 Attribute로 갖는다. 이 때 reorder _ID는 primary key로 1 이상의 양의 정수의 값을 가지는 고유한 재주문ID를 나타내어 각각의 reorder 정보를 구분한다.

quantity, reorder_date, promised_arrival_date, arrival_date는 각각 재주문 수량, 재주문 일자, 예정된 도착일자, 실제 도착일자를 나타내는 Attribute이다.

completed는 제품이 아직 도착하지 않았는지, 제품이 예정 시간에 맞게 도착했는지, 예정 시간보다 늦게 도착했는지, reorder 요청이 거부되었는지를 나타내는 Attribute이다. 아래의 표는 각각의 값에 따른 completed의 내용이다.

completed	
0	아직 제품이 도착하지 않은 경우
1	제품이 예정 시간에 맞게 도착한 경우
2	제품이 예정 시간보다 늦게 도착한 경우
3	reorder 요청이 거절된 경우

issue는 reorder 요청이 거절된 경우 거절 원인 등의 이슈 사항을 나타내는 Attribute이다. 예를 들어 해당 제품의 판매 중지로 reorder 요청이 거절되면 issue에

'stop_sale'를 저장한다.

1.17 online_reorder

- Entity 도출

주어진 고려사항에서 재고가 부족하면 제조사에 reorder를 보내고 reorder 리스트를 데이터베이스에 저장한다고 나와있으므로, 온라인 고객에게 배송되는 제품의 재고를 관리하는 창고에서 제조사에 보내는 reorder 정보를 저장하는 online_reorder Entity를 도출했다.

- Attributes 도출

online_reorder Entity는 reorder_ID, quantity, reorder_date, promised_arrival_date, arrival_date, completed, issue를 Attribute로 갖는다. 이 때 reorder_ID는 primary key로 1 이상의 양의 정수의 값을 가지는 고유한 재주문ID를 나타내어 각각의 reorder 정보를 구분한다.

quantity, reorder_date, promised_arrival_date, arrival_date는 각각 재주문 수량, 재주문 일자, 예정된 도착일자, 실제 도착일자를 나타내는 Attribute이다.

completed는 제품이 아직 도착하지 않았는지, 제품이 예정 시간에 맞게 도착했는지, 예정 시간보다 늦게 도착했는지, reorder 요청이 거부되었는지를 나타내는 Attribute이다. 아래의 표는 각각의 값에 따른 completed의 내용이다.

completed	
0	아직 제품이 도착하지 않은 경우
1	제품이 예정 시간에 맞게 도착한 경우
2	제품이 예정 시간보다 늦게 도착한 경우
3	reorder 요청이 거절된 경우

issue는 reorder 요청이 거절된 경우 거절 원인 등의 이슈 사항을 나타내는 Attribute이다. 예를 들어 해당 제품의 판매 중지로 reorder 요청이 거절되면 issue에 'stop_sale'를 저장한다.

1.18 manufacturer

- Entity 도출

주어진 고려사항에서 제품은 제조사(manufacturer)에 의해 분류될 수 있다고 나와 있고, 재고가 부족할 경우 reorder를 제조사에 보내야 한다고 나와있으므로, 제조사에 대한 정보를 저장하기 위해 manufacturer Entity를 도출했다.

- **Attributes 도출**

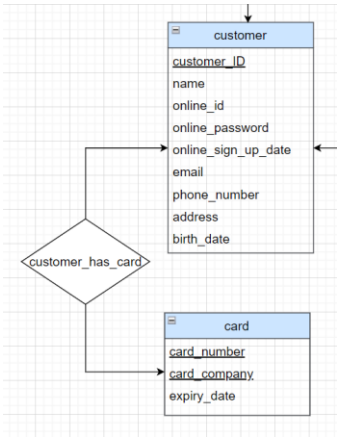
Manufacturer Entity는 manufacturer_ID, company_name, CEO, phone_number, address, email을 Attribute로 갖는다. 이 때 manufacturer_ID는 primary key로 1 이상의 양의 정수의 값을 가지는 고유한 제조사ID를 나타내어 각각의 제조사 정보를 구분한다.

company_name, CEO, phone_number, address, email은 각각 제조사 이름, CEO 이름, 전화번호, 제조사 주소, 이메일을 나타낸다.

2. Relationship & Mapping Cardinality

ER diagram을 생성하기 위해 위에서 설명한 Entity와 Attributes를 이용해서 각각의 entity들의 relationship과 mapping cardinality를 결정한다.

2.1 customer to card

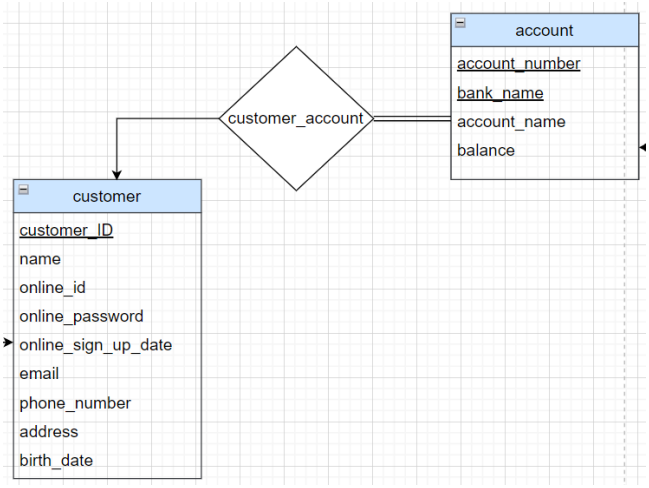


Relation name: customer_has_card

Cardinality: 1 : 1 Cardinality

주어진 고려사항에서 몇몇 고객은 신용카드나 직불카드로 지불한다고 나와있고, 온라인 고객의 카드 정보는 저장되어야 한다고 나와있으므로 고객정보와 카드정보 사이에 binary relationship을 설정하였다. 한 고객마다 한 개의 카드 정보만을 저장하므로 1 : 1 Cardinality를 갖는다.

2.2 customer to account

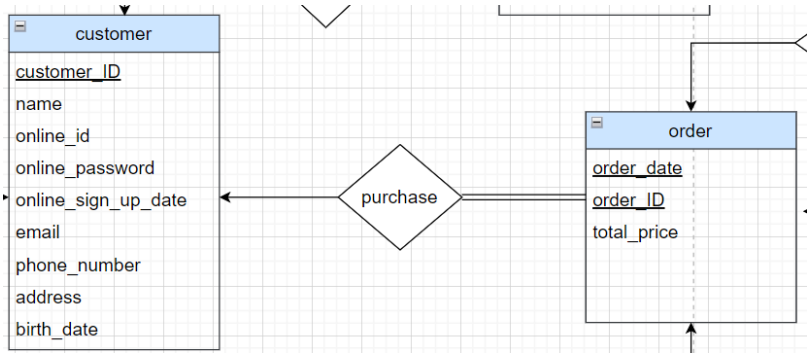


Relation name: customer_account

Cardinality: 1 : N Cardinality

주어진 고려사항에서 몇몇 고객은 구매 대금을 계좌 번호로 청구하며 달마다 청구한다고 나와 있으므로, 고객정보와 계좌정보 사이에 binary relationship을 설정하였다. 한 고객마다 여러 개의 계좌를 가질 수 있으므로 1 : N Cardinality를 갖는다. 또한 모든 계좌는 무조건 1명의 고객과 연관되어야 하므로 account Entity는 customer_account 관계에서 total participate한다.

2.3 customer to order

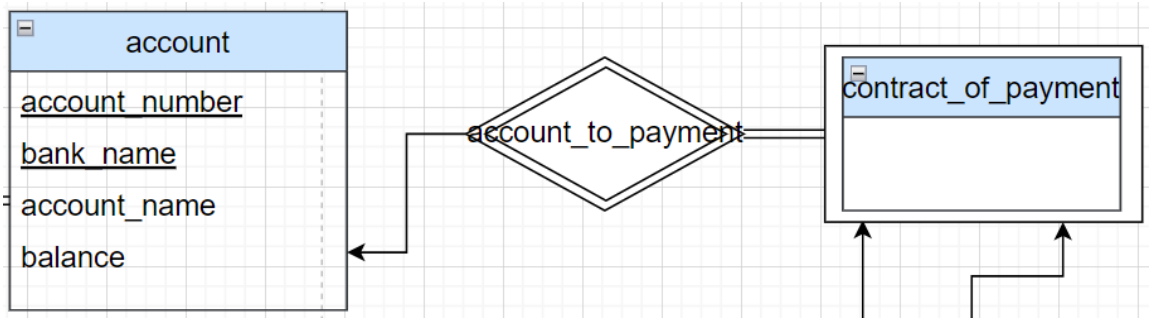


Relation name: purchase

Cardinality: 1 : N Cardinality

고객이 제품을 주문/구매를 할 경우 주문 정보를 관리할 수 있도록 고객정보와 주문정보 사이에 binary relationship을 설정하였다. 한 고객마다 여러 번의 주문을 할 수 있으므로 1 : N Cardinality를 갖는다. 또한 모든 주문은 무조건 1명의 고객과 연관되어야 하므로 order Entity는 purchase 관계에서 total participate한다.

2.4 account to contract_of_payment

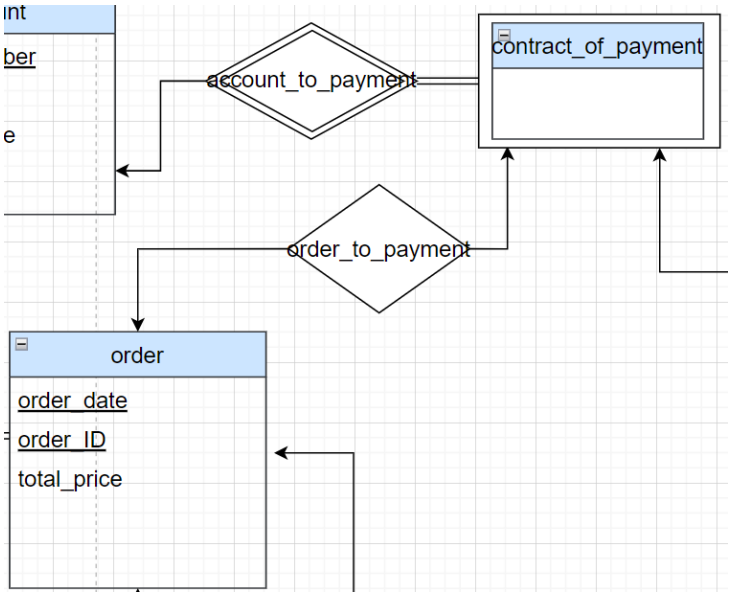


Relation name: account_to_payment

Cardinality: 1 : N Cardinality

주어진 고려사항에서 몇몇 고객은 구매 대금을 계좌 번호로 청구하며 달마다 청구한다고 나와 있으므로 계좌정보와 지불계약서 정보 사이에 binary relationship을 설정하였다. 한 계좌마다 여러 개의 지불계약서가 있을 수 있으므로 1 : N Cardinality를 갖는다. 또한 모든 지불계약서는 무조건 1개의 계좌와 연관되어야 하고 contract_of_payment Entity의 존재는 account Entity에 의존하기 때문에 contract_of_payment Entity는 weak Entity이고 account_to_payment는 identifying relationship이다. 따라서 contract_of_payment Entity는 account_to_payment 관계에서 total participate한다.

2.5 order to contract_of_payment

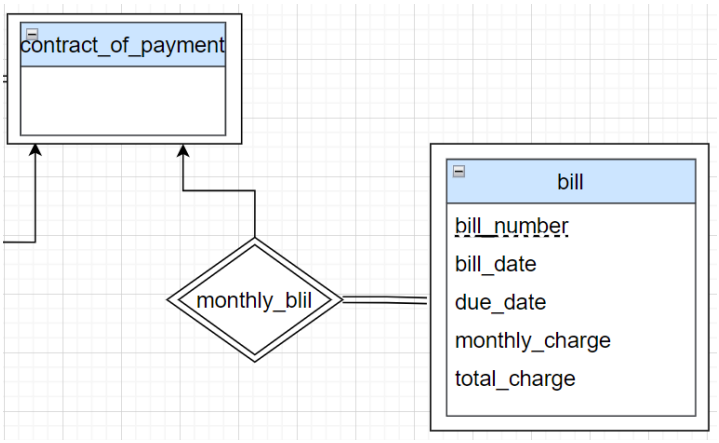


Relation name: order_to_payment

Cardinality: 1 : 1 Cardinality

주어진 고려사항에서 몇몇 고객은 구매 대금을 계좌 번호로 청구하며 달마다 청구한다고 나와 있으므로 주문/구매정보와 지불계약서 정보 사이에 binary relationship을 설정하였다. 한 개의 주문마다 최대 한 개의 지불계약서가 있을 수 있으므로 1 : 1 Cardinality를 갖는다.

2.6 contract_of_payment to bill

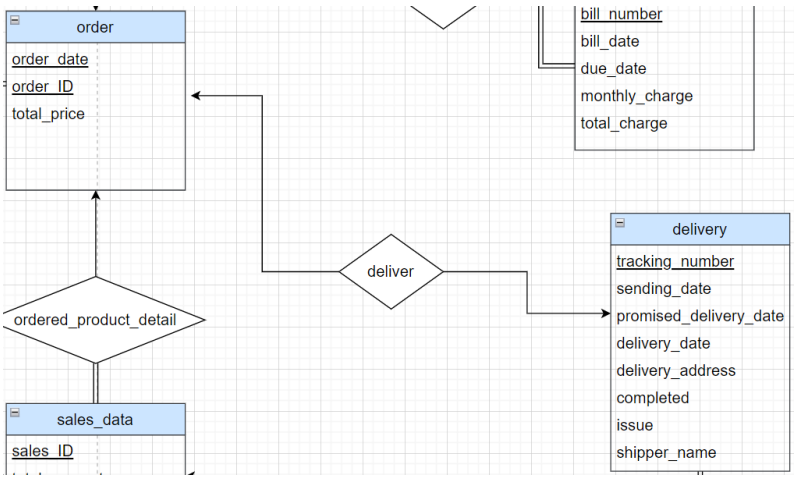


Relation name: monthly_bill

Cardinality: 1 : N Cardinality

주어진 고려사항에서 몇몇 고객은 구매 대금을 계좌 번호로 청구하며 달마다 청구한다고 나와 있으므로 지불계약서 정보와 청구서 정보 사이에 binary relationship을 설정하였다. 한 지불계약서마다 여러 개의 청구서가 있을 수 있으므로 1 : N Cardinality를 갖는다. 또한 모든 청구서는 무조건 1개의 지불계약서와 연관되어야 하고 bill Entity의 존재는 contract_of_payment Entity에 의존하기 때문에 bill Entity는 weak Entity이고 monthly_bill는 identifying relationship이다. 따라서 bill Entity는 monthly_bill 관계에서 total participate한다.

2.7 order to delivery



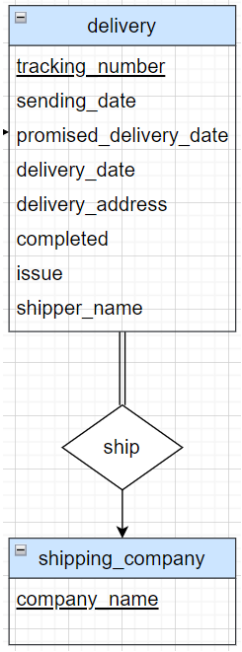
Relation name: deliver

Cardinality: 1 : 1 Cardinality

주어진 고려사항에서 online sale은 shipper를 통해서 전달되고 운송 회사에 대한 배송 추적 번호를 저장해야 한다고 나와 있으므로 주문 정보와 배송정보 사이에 binary relationship을

설정하였다. 한 개의 주문마다 최대 한 개의 배송정보가 있을 수 있으므로 1 : 1 Cardinality를 갖는다.

2.8 Shipping_company to delivery

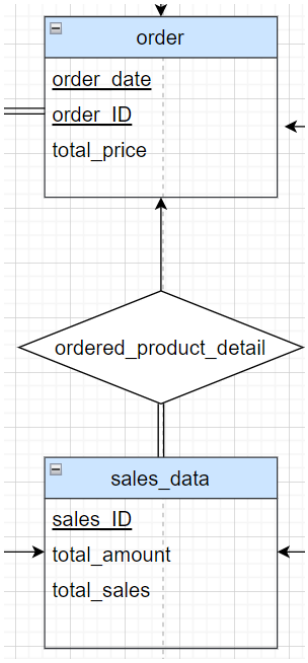


Relation name: ship

Cardinality: 1 : N Cardinality

주어진 고려사항에서 online sale은 shipper를 통해서 전달되고 운송회사에 대한 배송 추적 번호를 저장해야 한다고 나와 있으므로 운송회사 정보와 배송정보 사이에 binary relationship을 설정하였다. 한 운송회사마다 여러 개의 배송정보가 있을 수 있으므로 1 : N Cardinality를 갖는다. 또한 모든 배송정보는 무조건 1개의 운송회사와 연관되어야 하므로 delivery Entity는 ship 관계에서 total participate한다.

2.9 order to sales_data



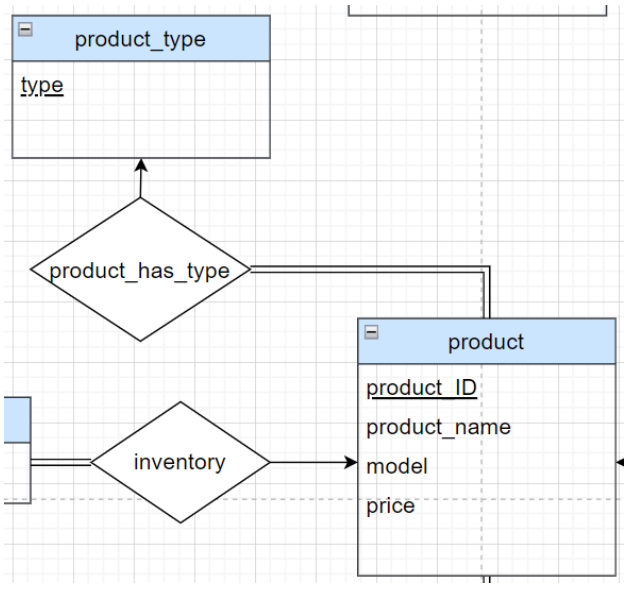
Relation name: ordered_product_detail

Cardinality: 1 : N Cardinality

고객이 제품을 주문했을 때 각 제품의 매출 정보를 관리하기 위해 주문 정보와 매출 정보 사이에 binary relationship을 설정하였다. 고객이 한 번 주문했을 때 여러 개의 제품이 판매될 수 있으므로 1 : N Cardinality를 갖는다. sales_data는 하나의 제품에 관한 매출정보이다

또한 모든 매출정보는 무조건 1개의 주문정보와 연관되어야 하므로 sales_data Entity는 ordered_product_detail 관계에서 total participate한다.

2.10 product_type to product

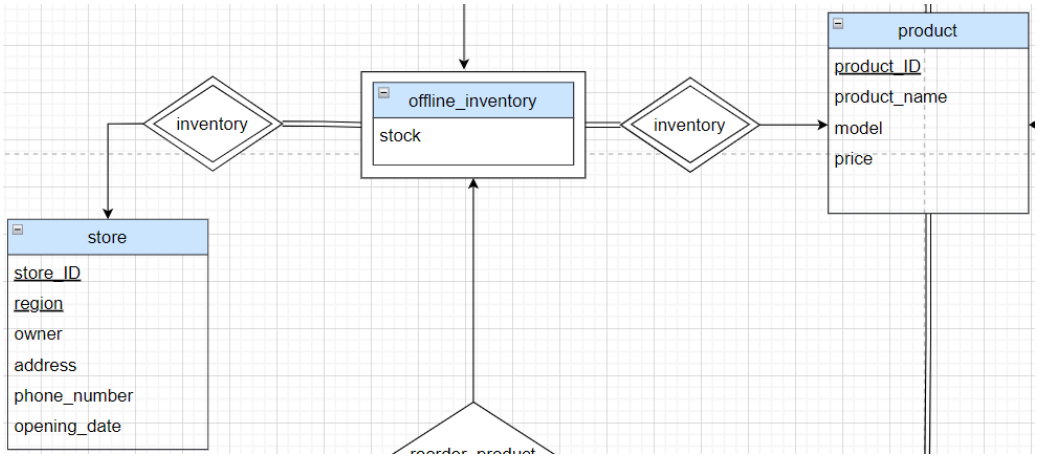


Relation name: product_has_type

Cardinality: 1 : N Cardinality

모든 제품은 제품 종류에 의해 분류될 수 있다. 따라서 제품 종류 정보와 제품 정보 사이에 binary relationship을 설정하였다. 하나의 제품 종류에 여러 개의 제품이 있을 수 있으므로 1 : N Cardinality를 갖는다. 또한 모든 제품은 무조건 1개의 제품 종류와 연관되어야 하므로 product Entity는 product_has_type 관계에서 total participate한다.

2.11 product to offline_inventory / store to offline_inventory



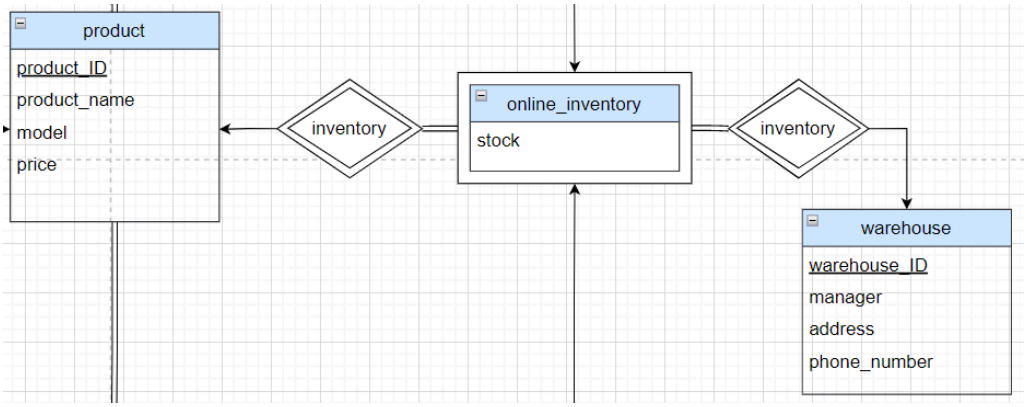
Relation name: inventory

Cardinality: 1 : N Cardinality

전자제품 판매 회사는 여러 개의 오프라인 상점(체인점)들을 운영하고 있다고 나와있으므로, 각 오프라인 상점에 있는 각 제품들의 재고 정보를 관리하기 위해 제품 정보와 오프라인 재고 정보 사이와, 상점 정보와 오프라인 재고 정보 사이에 binary relationship을 설정하였다. offline_inventory Entity는 하나의 상점에 있는 하나의 제품의 재고 정보이다.

하나의 제품에 여러 개의 제품 재고 정보가 있을 수 있고, 하나의 상점에 여러 개의 제품 재고 정보가 있을 수 있으므로 두 관계 모두 1 : N Cardinality를 갖는다. 또한 모든 제품 제고 정보는 무조건 1개의 제품과 상점에 연관되어야 하고 offline_inventory Entity의 존재는 store Entity와 product Entity에 의존하기 때문에 offline_inventory Entity는 weak Entity이고 inventory는 identifying relationship이다. 따라서 offline_inventory Entity는 inventory 관계에서 total participate한다.

2.12 product to online_inventory / store to online_inventory



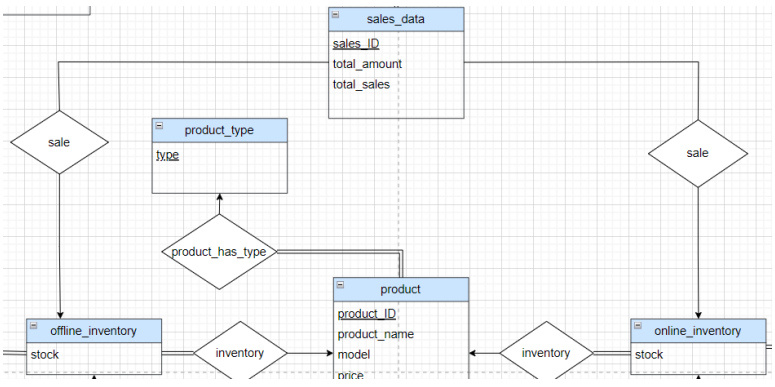
Relation name: inventory

Cardinality: 1 : N Cardinality

전자제품 판매 회사는 웹 사이트 또한 운영하고 있다고 나와있고, 주어진 고려사항에서 온라인 고객에게 배송되는 제품을 보관하는 창고를 관리한다고 나와있으므로, 각 창고에 있는 각 제품들의 재고 정보를 관리하기 위해 제품 정보와 온라인 재고 정보 사이와, 창고 정보와 온라인 재고 정보 사이에 binary relationship을 설정하였다. online_inventory Entity는 하나의 창고에 있는 하나의 제품의 재고 정보이다.

하나의 제품에 여러 개의 제품 재고 정보가 있을 수 있고, 하나의 창고에 여러 개의 제품 재고 정보가 있을 수 있으므로 두 관계 모두 1 : N Cardinality를 갖는다. 또한 모든 제품 제고 정보는 무조건 1개의 제품과 창고에 연관되어야 하고 online_inventory Entity의 존재는 warehouse Entity와 product Entity에 의존하기 때문에 online_inventory Entity는 weak Entity이고 inventory는 identifying relationship이다. 따라서 online_inventory Entity는 inventory 관계에서 total participate한다.

2.13 offline_inventory to sales_data / online_inventory to sales_data



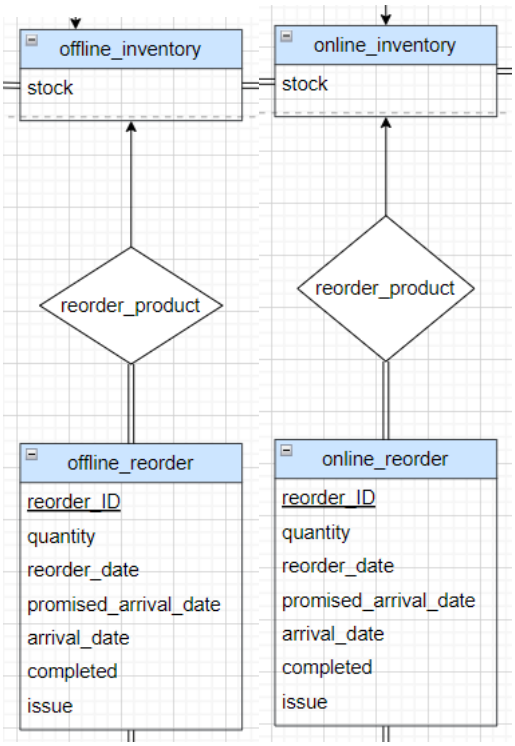
Relation name: sale

Cardinality: 1 : N Cardinality

주어진 고려사항에서 매출 정보는 기업 계획에 중요하고, 마케팅 담당자는 기간, 제품, 제품

분류, 시즌, 지역에 따른 매출 정보를 원한다고 나와있으므로, 오프라인 재고 정보와 매출 정보 사이와, 온라인 재고 정보와 매출 정보 사이에 binary relationship을 설정하였다. 하나의 상점/창고에 있는 제품은 여러 번 팔릴 수 있기 때문에 두 관계 모두 1 : N Cardinality를 갖는다.

2.14 offline_inventory to offline_reorder / online_inventory to online_reorder



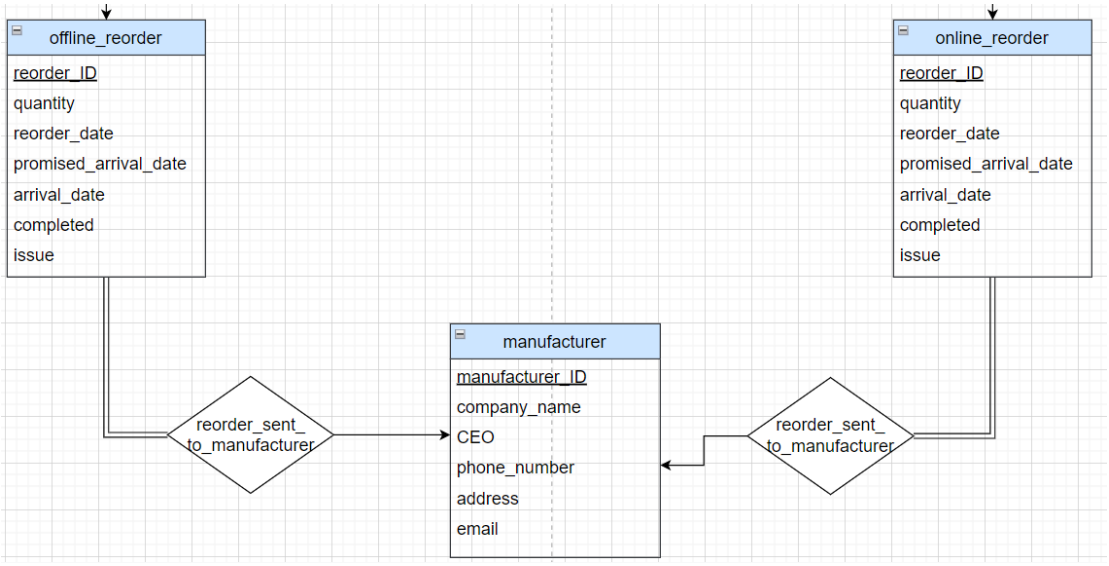
Relation name: reorder_product

Cardinality: 1 : N Cardinality

주어진 고려사항에서 재고가 부족하면 제조사에 reorder를 보내고 reorder 리스트를 데이터베이스에 저장한다고 나와있으므로, 오프라인 재고 정보와 오프라인 reorder 정보 사이와, 온라인 재고 정보와 온라인 reorder 정보 사이에 binary relationship을 설정하였다. 하나의 상점/창고에 있는 제품은 여러 번 reorder 할 수 있으므로 두 관계 모두 1 : N Cardinality를 갖는다.

또한 모든 reorder 정보는 무조건 1개의 재고 정보와 연관되어야 하므로 offline_reorder Entity와 online_reorder Entity는 reorder_product 관계에서 total participate한다

2.15 manufacturer to offline_reorder / manufacturer to online_reorder

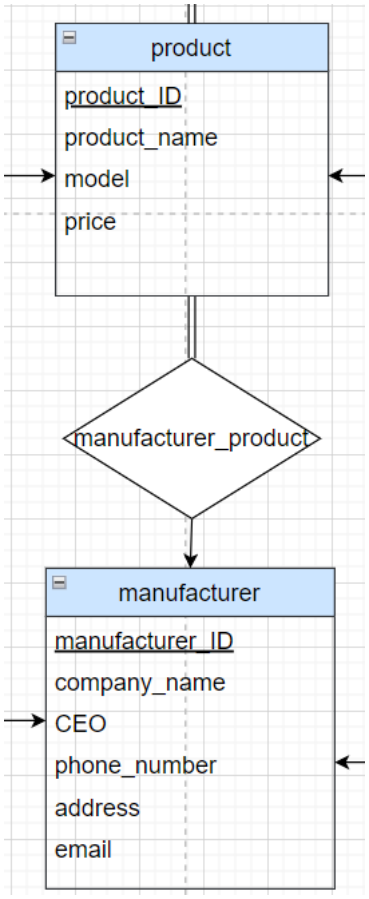


Relation name: reorder_sent_to_manufacturer

Cardinality: 1 : N Cardinality

주어진 고려사항에서 재고가 부족하면 제조사에 reorder를 보낸다고 나와있으므로, 오프라인 reorder 정보와 제조사 정보 사이와, 온라인 reorder 정보와 제조사 정보 사이에 binary relationship을 설정하였다. 하나의 제조사에 여러 개의 제품 reorder 정보가 있을 수 있으므로 두 관계 모두 1 : N Cardinality를 갖는다.

2.16 Manufacturer to product

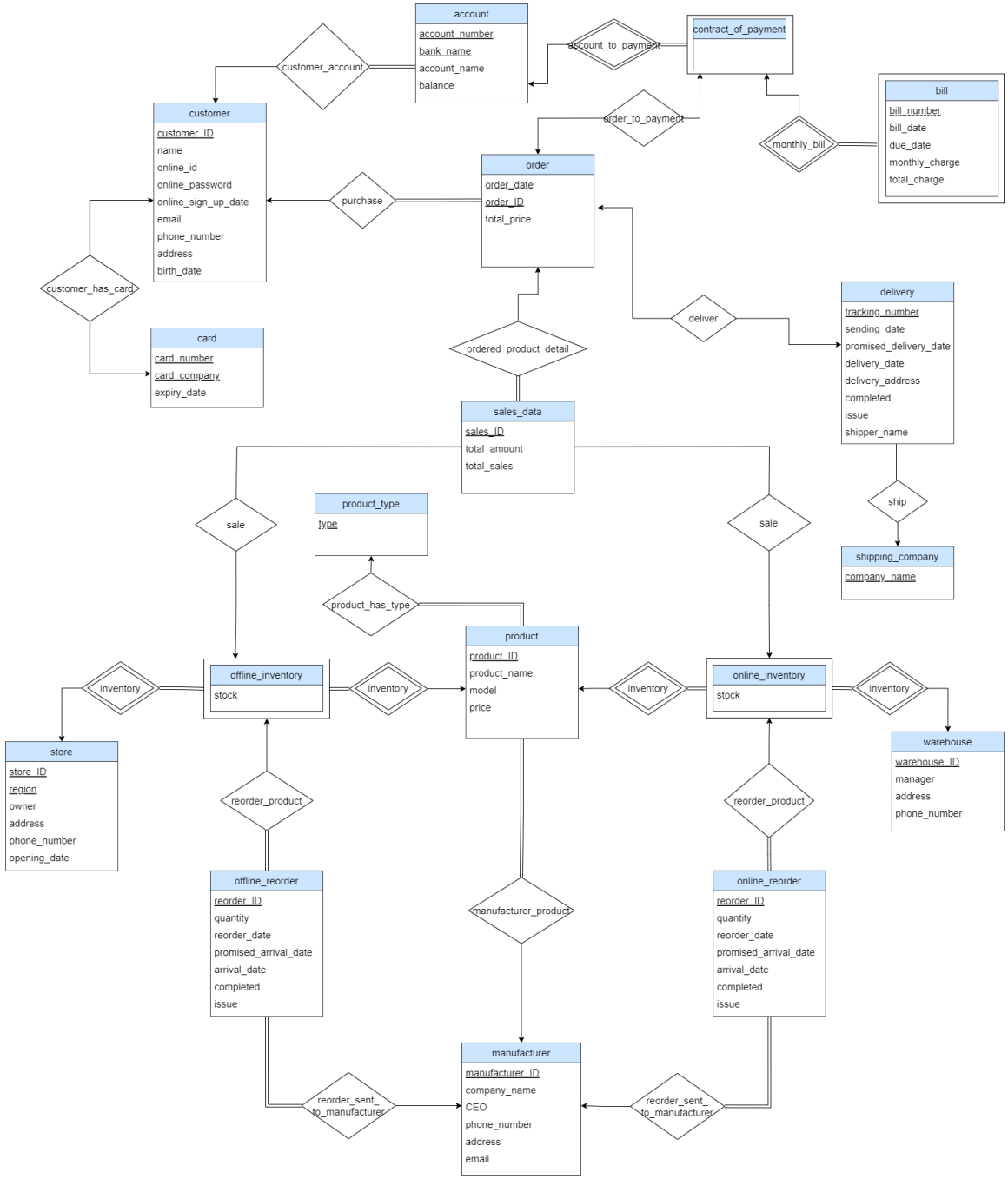


Relation name: manufacturer_product

Cardinality: 1 : N Cardinality

주어진 고려사항에서 제품은 제조사(manufacturer)에 의해 분류될 수 있다고 나와 있으므로, 제품 정보와 제조사 정보 사이에 binary relationship을 설정하였다. 하나의 제조사에 여러 개의 제품 정보가 있을 수 있으므로 두 관계 모두 1 : N Cardinality를 갖는다. 또한 모든 제품 정보는 무조건 1개의 제조사 정보와 연관되어야 하므로 product Entity는 manufacturer_product 관계에서 total participate한다.

2.17 전체 E-R 다이어그램

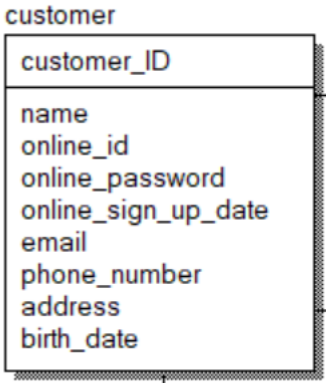


3. Relational Schema Diagram

생성한 E-R model을 바탕으로 Relational Schema Diagram을 생성한다.

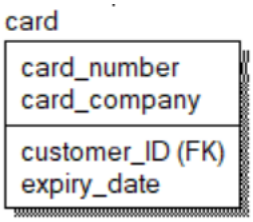
3.1 Entity

- **customer**



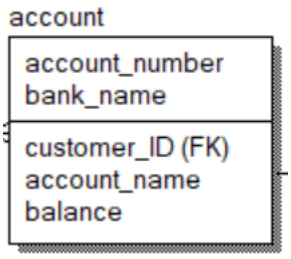
customer_ID	customer를 구분하는 primary key로 고유한 1 이상의 양의 정수의 값을 갖는다
name	customer의 이름 정보를 저장한다. (Not Null)
online_id	customer의 온라인 사이트 아이디 정보를 저장한다.
online_password	customer의 온라인 사이트 패스워드 정보를 저장한다.
online_sign_up_date	customer의 온라인 사이트 가입일자 정보를 저장한다.
email	customer의 이메일 정보를 저장한다.
phone_number	customer의 전화번호 정보를 저장한다.
address	customer의 주소 정보를 저장한다.
birth_date	customer의 생년월일 정보를 저장한다.

- **card**



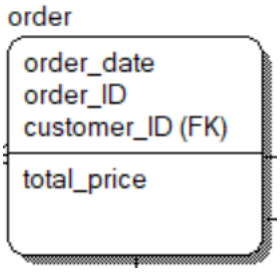
card_number	primary key로서 card의 카드 번호 정보를 저장한다.
card_company	primary key로서 card의 카드 회사 정보를 저장한다.
expiry_date	card의 유효기간 정보를 저장한다. (Not Null)

- **account**



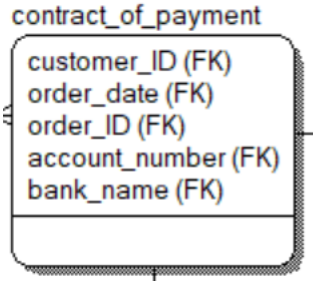
account_number	primary key로서 account의 계좌번호 정보를 저장한다.
bank_name	primary key로서 account의 은행 이름 정보를 저장한다.
account_name	account의 계좌명 정보를 저장한다.
balance	account의 잔액 정보를 저장한다. (Not Null)

- **order**

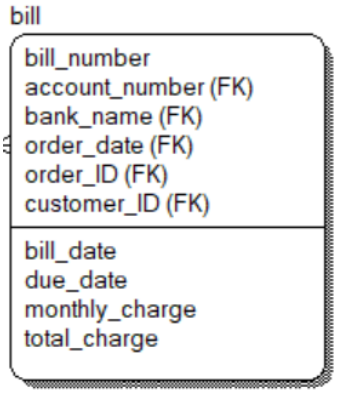


order_date	primary key로서 order의 주문/구매일자 정보를 저장한다
order_ID	order를 구분하는 primary key로 고유한 1 이상의 양의 정수의 값을 갖는다.
total_price	고객이 주문한(구매한) 모든 제품의 총 금액 정보를 저장한다. (Not Null)

- **contract_of_payment**



- **bill**

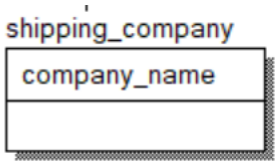


bill_number	primary key로 매달 생성되는 청구서에 대해 고유한 값을 가지는 청구서 번호 정보를 저장한다.
bill_date	bill의 청구일자 정보를 저장한다. (Not Null)
due_date	bill의 지불 만기일 정보를 저장한다. (Not Null)
monthly_charge	bill의 월 청구액 정보를 저장한다. (Not Null)
total_charge	bill의 총 청구액 정보를 저장한다. (Not Null)

- **delivery**

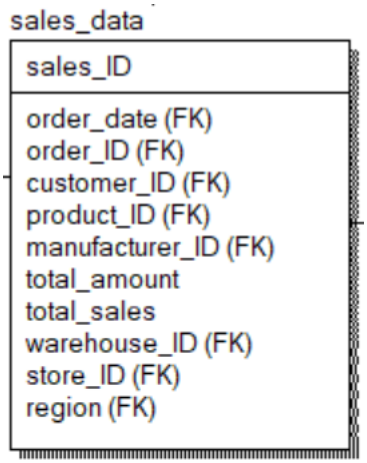
tracking_number	primary key로 고유한 값을 가지는 배송 추적번호 정보를 저장한다.
sending_date	delivery의 발송일자 정보를 저장한다. (Not Null)
promised_delivery_date	delivery의 예정 배송일자 정보를 저장한다. (Not Null)
delivery_date	delivery의 실제 배송일자 정보를 저장한다.
delivery_address	delivery의 배송주소 정보를 저장한다. (Not Null)
completed	delivery의 배송 완료 여부 정보를 저장한다. (Not Null)
issue	delivery의 이슈 정보를 저장한다.
shipper_name	delivery의 배송인 이름 정보를 저장한다. (Not Null)

- **shipping_company**



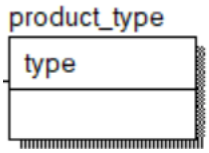
company_name	primary key로 shipping_company의 회사명 정보를 저장한다.

- **sales_data**



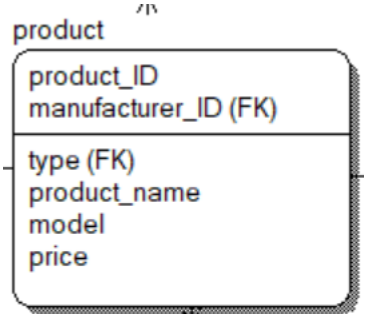
sales_ID	sales_data를 구분하는 primary key로 고유한 1 이상의 양의 정수의 값을 갖는다.
total_amount	각 제품이 주문/구매될 때의 총 수량 정보를 저장한다. (Not Null)
total_sales	각 제품이 주문/구매될 때의 총 판매액 정보를 저장한다. (Not Null)

- **product_type**



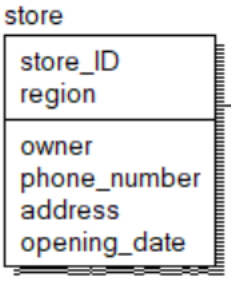
type	primary key로 product_type의 제품 종류명 정보를 저장한다.

- **product**



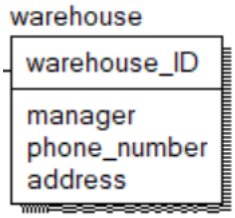
product_ID	product를 구분하는 primary key로 고유한 1 이상의 양의 정수의 값을 갖는다.
product_name	product의 이름 정보를 저장한다. (Not Null)
model	product의 모델명 정보를 저장한다.
price	product의 가격 정보를 저장한다. (Not Null)

- **store**



store_ID	지역별로 각각의 store를 구분하는 primary key로 고유한 1 이상의 양의 정수의 값을 갖는다.
region	primary key로 store의 지역 정보를 저장한다.
owner	store의 소유주 정보를 저장한다. (Not Null)
phone_number	store의 전화번호 정보를 저장한다. (Not Null)
address	store의 주소 정보를 저장한다. (Not Null)
opening_date	store의 개업일자 정보를 저장한다. (Not Null)

- **warehouse**



warehouse_ID	warehouse를 구분하는 primary key로 고유한 1 이상의 양의 정수의 값을 갖는다
manager	warehouse의 창고 관리인 이름 정보를 저장한다. (Not Null)
phone_number	warehouse의 전화번호 정보를 저장한다. (Not Null)
address	warehouse의 주소 정보를 저장한다. (Not Null)

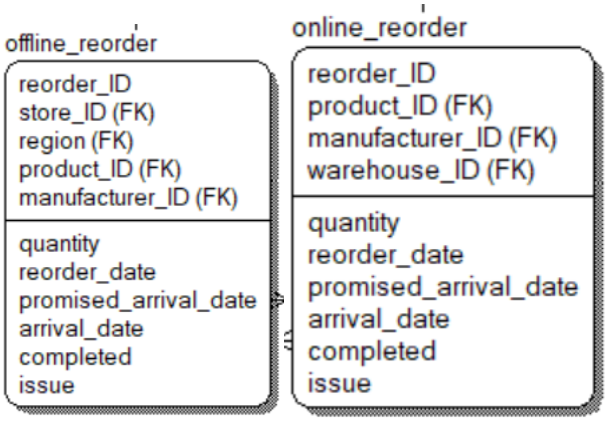
- **offline_inventory**

stock	각각의 상점에 있는 각각의 제품의 재고 수량 정보를 저장한다. (Not Null)

- **online_inventory**

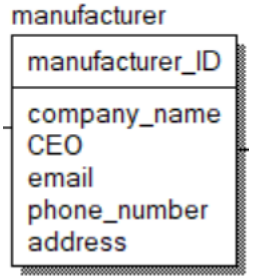
stock	각각의 상점에 있는 각각의 제품의 재고 수량 정보를 저장한다. (Not Null)

- **offline_reorder / online_reorder**



reorder_ID	reorder를 구분하는 primary key로 고유한 1 이상의 양의 정수의 값을 갖는다.
quantity	reorder의 재주문 수량 정보를 저장한다. (Not Null)
reorder_date	reorder의 재주문 일자 정보를 저장한다. (Not Null)
promised_arrival_date	reorder의 예정된 도착일자 정보를 저장한다. (Not Null)
arrival_date	reorder의 실제 도착일자 정보를 저장한다.
completed	reorder의 재주문 완료 여부 정보를 저장한다. (Not Null)
issue	reorder의 이슈 정보를 저장한다.

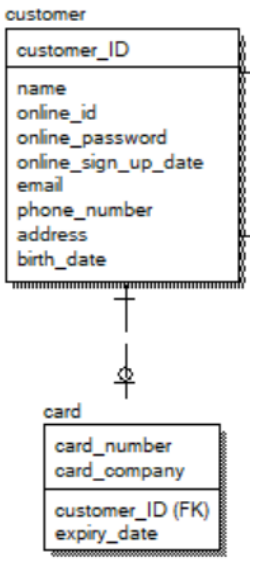
- **manufacturer**



manufacturer_ID	manufacturer를 구분하는 primary key로 고유한 1 이상의 양의 정수의 값을 갖는다.
company_name	manufacturer의 이름 정보를 저장한다. (Not Null)
CEO	manufacturer의 CEO 이름 정보를 저장한다. (Not Null)
email	manufacturer의 이메일 정보를 저장한다. (Not Null)
phone_number	manufacturer의 전화번호 정보를 저장한다. (Not Null)
address	manufacturer의 주소 정보를 저장한다. (Not Null)

3.2 Relation & Relationship Cardinality

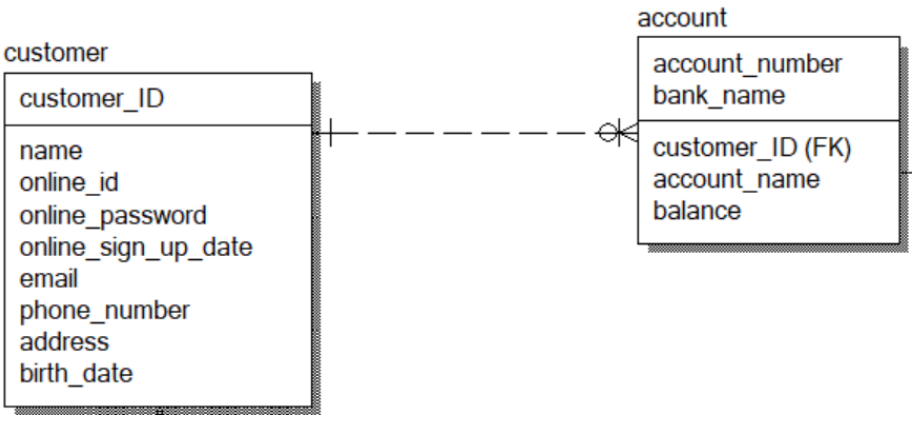
- **customer to card**



One to Zero or One / Non-Identifying / No Nulls

주어진 고려사항에서 몇몇 고객은 신용카드나 직불카드로 지불한다고 나와있고, 온라인 고객의 카드 정보는 저장될 수 있고 오프라인 상점 고객의 카드 정보는 저장하지 않는다고 나와있다. 즉, 고객 중 카드 정보가 없는 고객이 있을 수도 있으며 한 고객마다 한 개의 카드 정보만을 저장하므로, One to Zero or One Cardinality으로 설정하였다. 또한 card_number와 card_company의 조합만으로 Entity를 구분할 수 있기 때문에 Non-Identifying 관계로 설정하였으며, 카드 정보는 항상 고객 정보를 가져야 하기 때문에 No Nulls로 설정하였다.

- **customer to account**

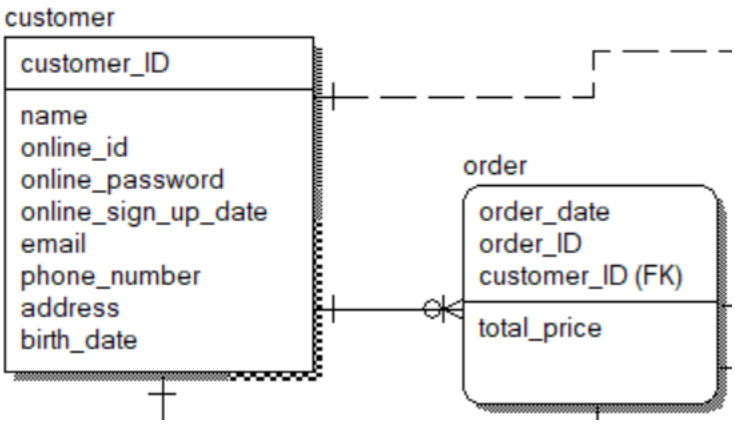


One to Zero, One or More / Non-Identifying / No Nulls

주어진 고려사항에서 몇몇 고객은 구매 대금을 계좌 번호로 청구하며 달마다 청구한다고 나와 있다. 한 고객은 여러 개의 계좌를 가지거나 아예 안 가질 수 있으므로

One to Zero, One or More Cardinality으로 설정하였다. 또한 account_number와 bank_name의 조합만으로 Entity를 구분할 수 있기 때문에 Non-Identifying 관계로 설정하였으며, 계좌 정보는 항상 고객 정보를 가져야 하기 때문에 No Nulls로 설정하였다.

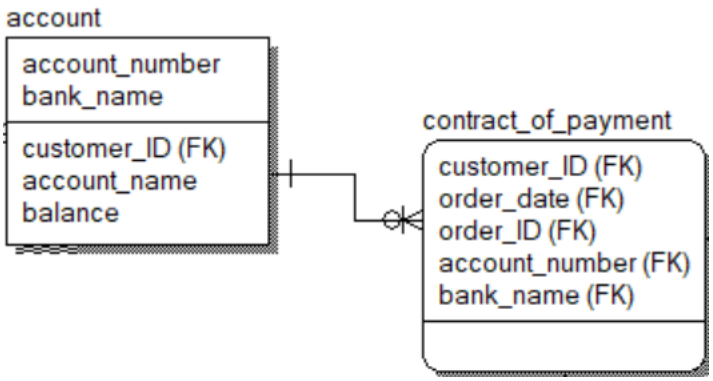
- **customer to order**



One to Zero, One or More / Identifying / No Nulls

고객은 여러 번의 주문/구매를 할 수 있고, 온라인 사이트 등을 통해서 회원가입을 하였지만 아직 주문을 하지 않은 고객이 있을 수도 있으므로 One to Zero, One or More Cardinality으로 설정하였다. 또한 order_date, order_ID와 Foreign key로 들어온 customer_ID 와 조합되어야만 Entity를 구분할 수 있도록 하기 위해 Identifying 관계로 설정하였다. 그리고 주문/구매 시 발생한 order에 대하여 항상 고객정보를 가져야 하므로 Null을 허용하지 않는다.

- **account to contract_of_payment**

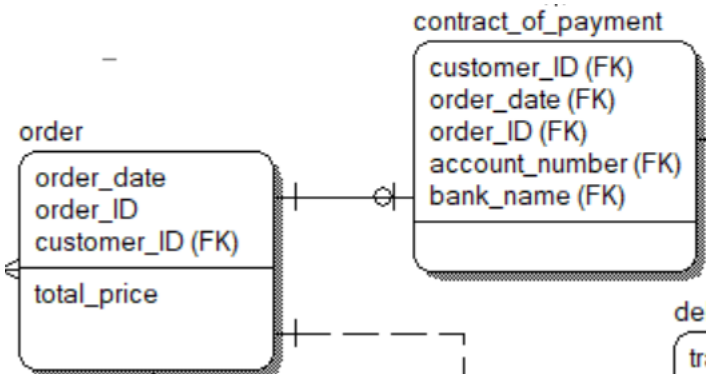


One to Zero, One or More / Identifying / No Nulls

주어진 고려사항에서 몇몇 고객은 구매 대금을 계좌 번호로 청구하며 달마다 청구한다고 나와 있으므로 contract_of_payment(지불 계약서) Entity와 account Entity 사이에 관계를 설정한다. 한 계좌마다 여러 개의 지불계약서가 있을 수 있고 아직 한 계좌로

지불대금을 청구하지 않을 수도 있으므로 One to Zero, One or More Cardinality으로 설정하였다. 또한 contract_of_payment Entity의 primary key는 account Entity와 order Entity의 primary key를 foreign key로 받아 이들의 조합을 primary key로 설정하기 때문에 Identifying 관계이며 Null을 허용하지 않는다.

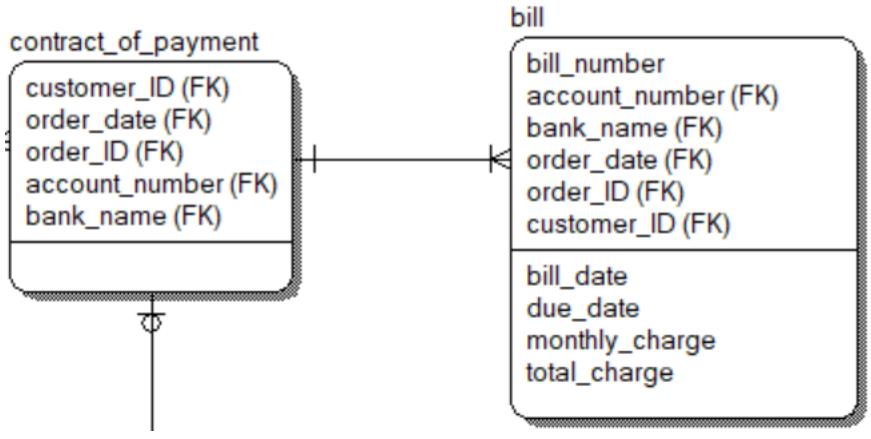
- **order to contract_of_payment**



One to Zero, One or More / Identifying / No Nulls

주어진 고려사항에서 몇몇 고객은 구매 대금을 계좌 번호로 청구하며 달마다 청구한다고 나와 있으므로 contract_of_payment(지불 계약서) Entity와 order Entity 사이에 관계를 설정한다. 한 개의 주문마다 최대 한 개의 지불계약서가 있을 수 있고, 한 개의 주문에 대해 계좌로 지불대금을 청구하지 않고 다른 방법으로 지불할 수도 있으므로 One to Zero or One Cardinality으로 설정하였다. 또한 contract_of_payment Entity의 primary key는 account Entity와 order Entity의 primary key를 foreign key로 받아 이들의 조합을 primary key로 설정하기 때문에 Identifying 관계이며 Null을 허용하지 않는다.

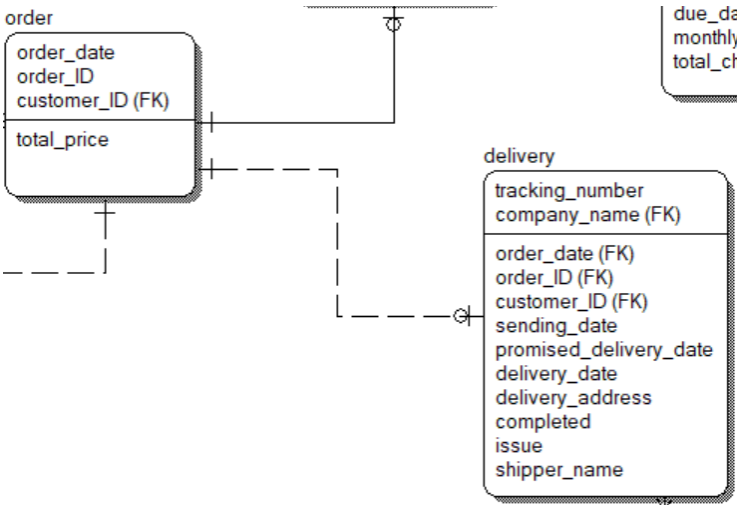
- **contract_of_payment to bill**



One to One or More / Identifying / No Nulls

주어진 고려사항에서 몇몇 고객은 구매 대금을 계좌 번호로 청구하며 달마다 청구한다고 나와 있다. 즉, 한 지불계약서마다 최소 1개 이상의 청구서가 있으므로 One to One or More Cardinality으로 설정하였다. 또한 bill number와 Foreign key로 들어온 contract_of_payment Entity의 primary key와 조합 되어야만 bill Entity를 구분할 수 있기 때문에 Identifying 관계로 설정하였다. 그리고 청구서 정보는 항상 지불계약서 정보를 무조건 가져야 하기 때문에 Null을 허용하지 않는다.

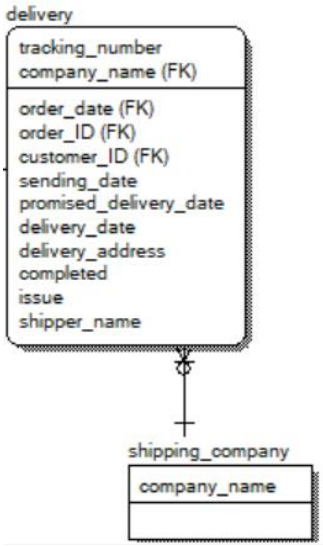
- **order to delivery**



One to Zero or One / Non-Identifying / No Nulls

주어진 고려사항에서 online sale은 shipper를 통해서 전달되고 운송 회사에 대한 배송 추적 번호를 저장해야 한다고 나와있다. 즉, 몇몇 주문/구매는 배송 정보가 없을 수도 있으며 없는 한 주문마다 오직 한 개의 배송 정보만을 저장하므로, One to Zero or One Cardinality으로 설정하였다. 또한 tracking_number와 company_name의 조합만으로 Entity를 구분할 수 있기 때문에 Non-Identifying 관계로 설정하였으며, 배송 정보는 항상 주문 정보를 가져야 하기 때문에 No Nulls로 설정하였다.

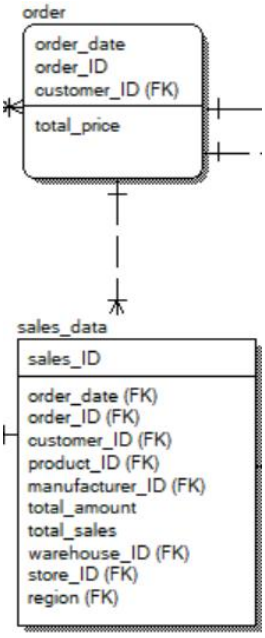
- **shipping_company to delivery**



One to Zero, One or More / Identifying / No Nulls

주어진 고려사항에서 online sale은 shipper를 통해서 전달되고 운송 회사에 대한 배송 추적 번호를 저장해야 한다고 나와있다. 한 운송회사마다 여러 개의 배송정보가 있을 수 있으므로, One to Zero, One or More Cardinality으로 설정하였다. 또한 tracking_number와 shipping_company의 primary key의 조합으로 delivery Entity를 구분할 수 있기 때문에 Identifying 관계로 설정하였으며, 배송 정보는 항상 운송회사 정보를 가져야 하기 때문에 Null을 허용하지 않는다.

- **order to sales_data**

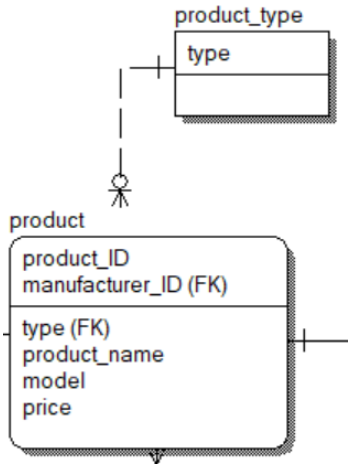


One to One or More / Non-Identifying / No Nulls

고객이 제품을 주문했을 때 각 제품의 매출 정보를 관리하기 위해 order Entity와

sales_data Entity 사이에 관계를 설정한다. 고객이 한 번 주문했을 때 최소 1개 이상의 제품이 판매될 수 있으므로 One to One or More Cardinality를 갖는다. sales_data는 하나의 제품에 관한 매출정보이다. 또한 order Entity의 primary key와 별도로 sales_ID를 생성하여 sales_data를 식별하므로 Non-Identifying이며 매출 정보는 항상 주문정보를 가져야 하기 때문에 No Nulls로 설정하였다.

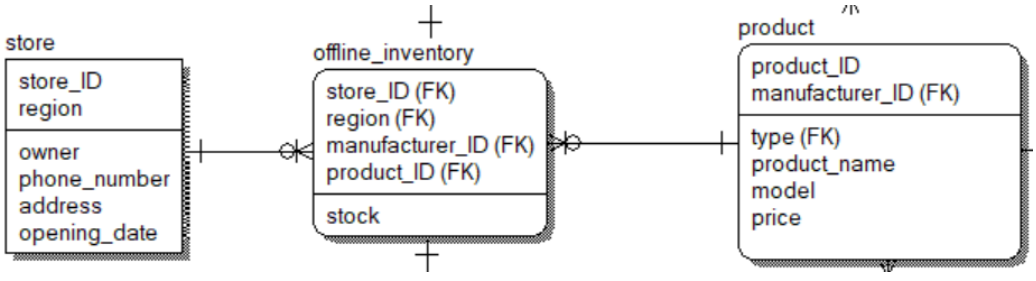
- **product_type to product**



One to Zero, One or More / Non-Identifying / No Nulls

주어진 고려사항에서 모든 제품은 제품 종류에 의해 분류될 수 있다고 나와있다. 하나의 제품 종류에 여러 개의 제품이 있을 수 있으므로 One to Zero, One or More Cardinality으로 설정하였다. 또한 product_ID와 manufacturer_ID의 조합만으로 product Entity를 구분할 수 있기 때문에 Non-Identifying 관계로 설정하였으며, 제품 정보는 항상 제품 종류 정보를 가져야 하기 때문에 No Nulls로 설정하였다.

- **product to offline_inventory / store to offline_inventory**

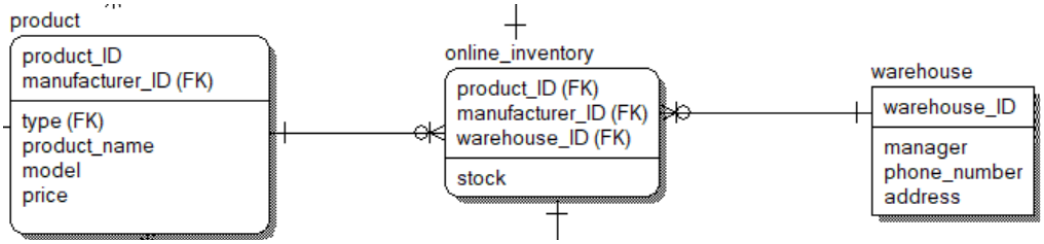


One to Zero, One or More / Identifying / No Nulls

전자제품 판매 회사는 여러 개의 오프라인 상점(체인점)들을 운영하고 있다고 나와있으므로, 각 오프라인 상점에 있는 각 제품들의 재고 정보를 관리하기 위해 product Entity와 offline_inventory Entity 사이와, store Entity와 offline_inventory 사이에 관계를 설정한다.

하나의 제품에 여러 개의 제품 재고 정보가 있거나 아예 없을 수도 있고, 하나의 상점에 여러 개의 제품 재고 정보가 있거나 아예 없을 수도 있으므로 두 관계 모두 One to Zero, One or More Cardinality으로 설정하였다. 또한 product Entity와 store Entity의 primary key들의 조합으로 offline_inventory Entity를 구분하기 때문에 두 관계 모두 Identifying 관계로 설정하였으며, 오프라인 제품 재고 정보는 항상 제품 정보와 상점 정보를 가져야 하기 때문에 Null을 허용하지 않는다.

- **product to online_inventory / warehouse to online_inventory**

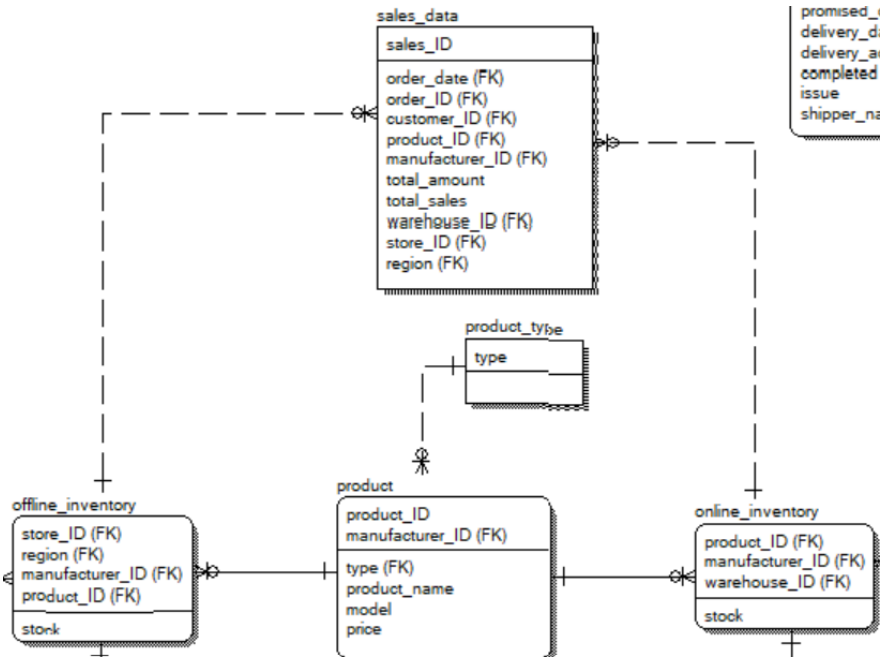


One to Zero, One or More / Identifying / No Nulls

전자제품 판매 회사는 웹 사이트 또한 운영하고 있다고 나와있고, 주어진 고려사항에서 온라인 고객에게 배송되는 제품을 보관하는 창고를 관리한다고 나와있으므로, 각 창고에 있는 각 제품들의 재고 정보를 관리하기 위해 product Entity와 online_inventory Entity 사이와, warehouse Entity와 online_inventory 사이에 관계를 설정한다.

하나의 제품에 여러 개의 제품 재고 정보가 있거나 아예 없을 수도 있고, 하나의 창고에 여러 개의 제품 재고 정보가 있거나 아예 없을 수도 있으므로 두 관계 모두 One to Zero, One or More Cardinality으로 설정하였다. 또한 product Entity와 warehouse Entity의 primary key들의 조합으로 online_inventory Entity를 구분하기 때문에 두 관계 모두 Identifying 관계로 설정하였으며, 온라인 제품 재고 정보는 항상 제품 정보와 창고 정보를 가져야 하기 때문에 Null을 허용하지 않는다.

- **offline_inventory to sales_data / online_inventory to sales_data**

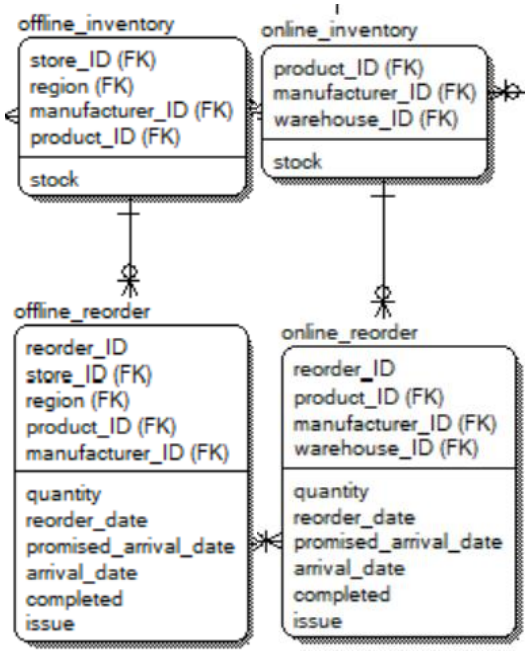


One to Zero, One or More / Non-Identifying / Nulls allowed

주어진 고려사항에서 매출 정보는 기업 계획에 중요하고, 마케팅 담당자는 기간, 제품, 제품 분류, 시즌, 지역에 따른 매출 정보를 원한다고 나와있으므로, offline_inventory Entity와 sales_data Entity 사이와, online_inventory Entity와 sales_data Entity 사이에 관계를 설정하였다. 하나의 상점/창고에 있는 제품은 여러 번 팔리거나 아예 안 팔릴 수 있기 때문에 두 관계 모두 One to Zero, One or More Cardinality를 갖는다.

또한 offline_inventory Entity와 online_inventory Entity의 primary key와 별도로 sales_ID를 생성하여 sales_data를 식별하므로 두 관계 모두 Non-Identifying이며, 하나의 매출 정보는 오프라인 또는 온라인 재고 정보 둘 중 하나의 정보만을 가지므로 두 관계 모두 Nulls allowed로 설정하였다.

- **offline_inventory to offline_reorder / online_inventory to online_reorder**

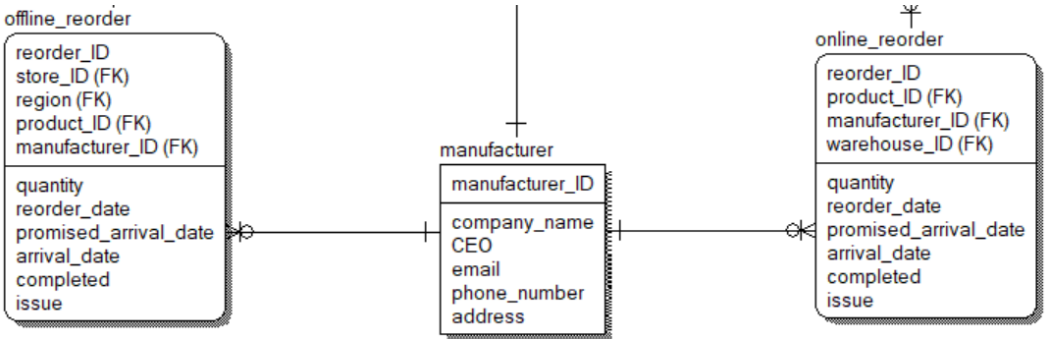


One to Zero, One or More / Identifying / No Nulls

주어진 고려사항에서 재고가 부족하면 제조사에 reorder를 보내고 reorder 리스트를 데이터베이스에 저장한다고 나와있으므로, 하나의 상점/창고에 있는 제품은 여러 번 reorder 하거나 아예 reorder하지 않을 수 있으므로 두 관계 모두 One to Zero, One or More Cardinality를 갖는다.

또한 reorder_ID와 각 inventory Entity의 primary key의 조합으로 각 reorder Entity를 구분하기 때문에 두 관계 모두 Identifying 관계로 설정하였으며, reorder 정보는 항상 제품 재고 정보를 가져야 하기 때문에 Null을 허용하지 않는다.

- **manufacturer to offline_reorder / manufacturer to online_reorder**

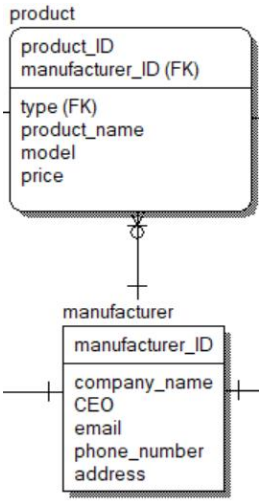


One to Zero, One or More / Identifying / No Nulls

주어진 고려사항에서 재고가 부족하면 제조사에 reorder를 보낸다고 나와있고, 하나의 제조사에 여러 번 reorder할 수 있거나 아예 안 할 수 있으므로 두 관계 모두 One to Zero, One or More Cardinality를 갖는다. 또한 manufacturer의 primary key인

manufacturer_ID가 각 reorder Entity를 구분하는데 사용되므로 두 관계 모두 Identifying 관계로 설정하였으며, reorder 정보는 항상 제조사 정보를 가져야 하기 때문에 Null을 허용하지 않는다.

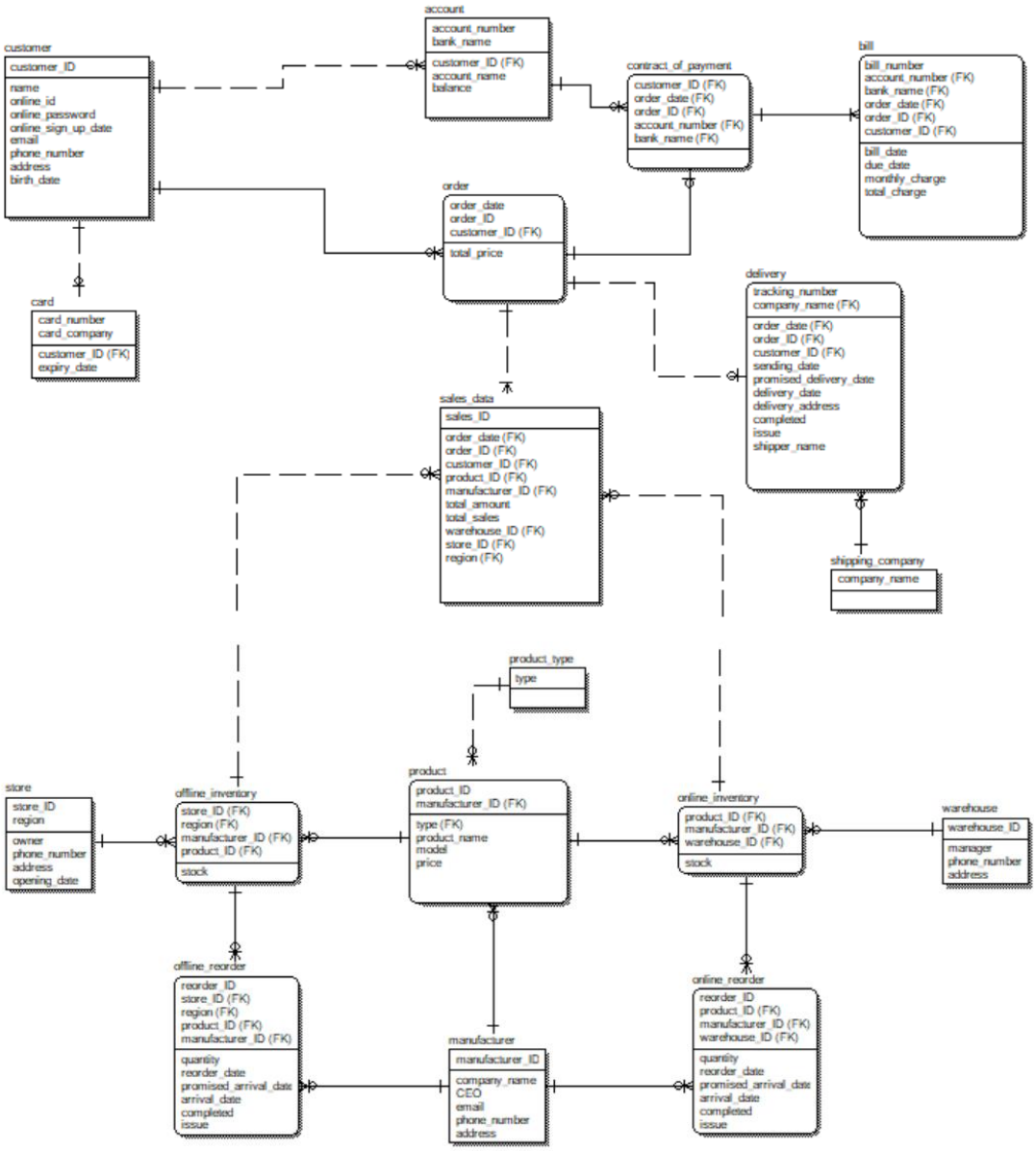
- **manufacturer to product**



One to Zero, One or More / Identifying / No Nulls

주어진 고려사항에서 제품은 제조사(manufacturer)에 의해 분류될 수 있다고 나와 있고, 하나의 제조사에 여러 개의 제품 정보가 있거나 아예 없을 수 있으므로 One to Zero, One or More Cardinality를 갖는다. 또한 product_ID와 manufacturer의 primary key의 조합으로 product Entity를 구분하기 때문에 Identifying으로 설정하였으며 제품 정보는 항상 제조사 정보를 가져야 하기 때문에 Null을 허용하지 않는다.

3.3 전체 구현 모습


customer
customer_ID
name
online_id
online_password
online_sign_up_date
email
phone_number
address
birth_date
account
account_number
bank_name
customer_ID (FK)
account_name
balance
contract_of_payment
customer_ID (FK)
order_date (FK)
order_ID (FK)
account_number (FK)
bank_name (FK)
bill
bill_number
account_number (FK)
bank_name (FK)
order_date (FK)
order_ID (FK)
customer_ID (FK)
bill_date
due_date
monthly_charge
total_charge
order
order_date
order_ID
customer_ID (FK)
total_price
card
card_number
card_company
customer_ID (FK)
expiry_date
delivery
tracking_number
company_name (FK)
order_date (FK)
order_ID (FK)
customer_ID (FK)
sending_date
promised_delivery_date
delivery_date
delivery_address
completed
issue
shipper_name
sales_data
sales_ID
order_date (FK)
order_ID (FK)
customer_ID (FK)
product_ID (FK)
manufacturer_ID (FK)
total_amount
total_sales
warehouse_ID (FK)
store_ID (FK)
region (FK)
shipping_company
company_name
product_type
type
store
store_ID
region
owner
phone_number
address
opening_date
offline_inventory
store_ID (FK)
region (FK)
manufacturer_ID (FK)
product_ID (FK)
stock
product
product_ID
manufacturer_ID (FK)
type (FK)
product_name
model
price
online_inventory
product_ID (FK)
manufacturer_ID (FK)
warehouse_ID (FK)
stock
warehouse
warehouse_ID
manager
phone_number
address
offline_reorder
reorder_ID
store_ID (FK)
region (FK)
product_ID (FK)
manufacturer_ID (FK)
quantity
reorder_date
promised_arrival_date
arrival_date
completed
issue
manufacturer
manufacturer_ID
company_name
CEO
email
phone_number
address
online_reorder
reorder_ID
product_ID (FK)
manufacturer_ID (FK)
warehouse_ID (FK)
quantity
reorder_date
promised_arrival_date
arrival_date
completed
issue