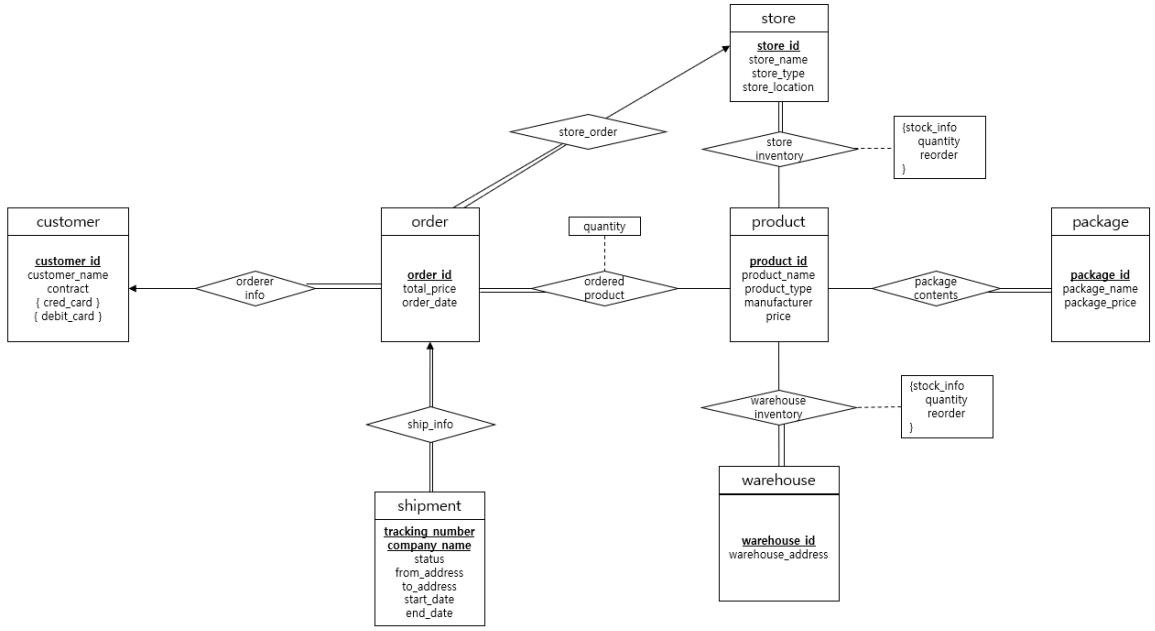
CSE4110 – Database System Project 1

E-R design and Relational Schema design

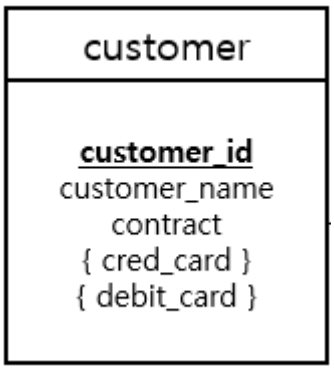
20171645 박찬우

1. E-R Model



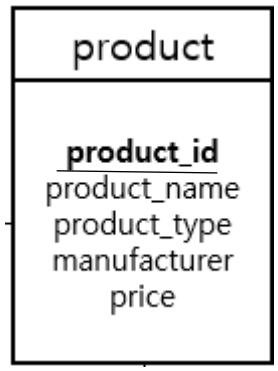
2. E-R Model Entity

(1) Customer



고객에 대한 정보를 저장하는 Entity이다. 각 손님을 구별할 수 있도록 주어지는 ID 넘버인 customer_id, 고객의 이름을 저장하는 customer_name, 정기적인 고객이 회사와 계약하여 요금이 달마다 청구되는지 여부를 저장하는 contract, 온라인 고객의 신용카드 정보를 저장하는 cred_card 와 온라인 고객의 직불카드 정보를 저장하는 debit_card 총 5가지의 Attribute로 구성되어 있다. 이때 각 손님을 customer_id를 통하여 식별할 수 있으므로 customer Entity의 primary key는 customer_id로 설정한다. cred_card 와 debit_card는 한 고객이 여러 카드를 등록할 수 있으므로 둘 다 multivalued attribute로 설정한다. 또한, 신용카드와 직불카드 둘 다 아예 등록하지 않을 수 있으므로, nullable하다. 이렇게 구성된 customer Entity를 통하여 고객의 개인정보를 입력, 수정, 삭제, 조회할 수 있게 된다.

(2) product

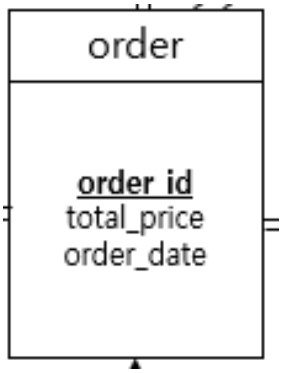


제품에 대한 정보를 저장하는 Entity이다. 각 제품을 구별할 수 있도록 주어지는 ID 넘버인 product_id, 제품의 이름을 저장하는 product_name, 제품이 어떤 종류의 제품인지, 예를들어 제품이 카메라인지 핸드폰인지 TV인지 컴퓨터인지 등, 에 대한 정보를 저장하는 product_type, 해당 제품을 만든 제조사 정보를 저장하는 manufacturer, 그리고 해당 제품의 개별 가격을 저장하는 price로 총 5가지의 Attribute를 갖는다. 이때 각 제품을 product_id를 통해 구별할 수 있으므로, product Entity의 primary key는 product_id로 설정한다. 이 Entity를 통하여 제품의 정보를 입력, 수정, 삭제, 조회할 수 있게 된다.

(3) Shipment

배송에 관련된 정보를 저장하는 Entity이다. 배송 각각에 붙는 id 역할을 하는 tracking_number, 배송회사의 이름을 저장하는 company_name, 배송준비중, 배송중, 배송완료 등으로 배송의 상태를 나타내는 status, 배송 출발지 정보를 저장하는 from_address, 배송 목적지 정보를 저장하는 to_address, 배송 시작일 정보를 저장하는 start_date, 배송 완료일 정보를 저장하는 end_date 로 총 7개의 Attribute로 구성되어있다. tracking number, 우리나라로 치면 운송장 번호 같은 경우 회사에 따라 책정방식이 다르고, 따라서 서로 다른 회사에서 tracking number가 같을 경우가 생길 수 있으므로, shipment의 primary key는 tracking_number와 company_name 두개의 attribute로 설정한다. 다른 Attribute와 달리 end_date Attribute는 아직 배송이 끝나지 않았으면 null값이 들어갈 수 있다. 이 Entity를 통하여 제품이 배송되는 정보를 입력, 수정, 삭제, 조회할 수 있게 된다.

(4) Order



주문에 대한 정보를 저장하는 Entity로, 개별 주문을 구분할 수 있도록 주문마다 주어지는 order_id, 주문의 총 금액을 나타내는 total_price와 주문이 일어난 날짜 정보를 저장하는 order_date 3가지 Attribute로 구성되어 있다. 각 주문은 order_id로 서로 구별 가능하고, 때문에 order Entity의 Primary key는 order_id 로 설정했다. 이 order Entity는 customer Entity, product Entity, shipment Entity 각각과 relation을 가져 이를 통해 주문 정보로부터 주문을 한 고객에 대한 정보, 주문한 제품에 대한 정보, 주문의 배송과 관련된 정보에 접근할 수 있게 된다. 이에 대한 자세한 설명은 아래 Relation 설명에서 이어진다.

이 Entity를 통하여 주문에 대한 정보를 입력, 수정, 삭제, 조회할 수 있게 된다.

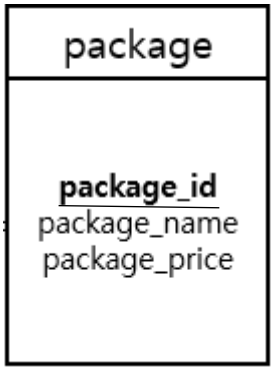
(5) Warehouse

창고에 대한 정보를 저장하는 Entity로, 각각의 창고에 붙어 서로를 구별하는 warehouse_id 와 창고의 위치 정보를 저장하는 warehouse_address 2가지 Attribute로 되어있다. warehouse_id 를 통해 각 창고를 서로 구별할 수 있기 때문에, warehouse Entity의 primary key는 warehouse_id 로 설정했다. product Entity와 Relation를 형성하며, 해당 Relation에 stock Attribute을 추가하여 특정 창고에 어떤 제품이 얼마나 있는지에 대한 정보를 쉽게 얻을 수 있도록 한다. 이 Entity를 통하여 창고에 대한 정보를 입력, 수정, 삭제, 조회할 수 있게 된다.

(6) Store

매장에 대한 정보를 저장하는 Entity로, 각 매장에 다르게 붙는 번호인 store_id, 매장의 이름을 저장하는 store_name, 매장이 물리적 매장인지 온라인 매장인지에 대한 정보를 저장하는 store_type, 매장의 위치에 대한 정보를 저장하는 store_location 4가지 Attribute로 구성된다. 온라인 매장의 경우 store_location에 온라인 매장 도메인 주소를 넣어주도록 한다. 각 매장은 store_id를 통해 서로 구별 가능하므로 store Entity의 primary key는 store_id로 설정한다. warehouse와 비슷하게, product Entity와의 Relation를 형성, 그 Relation에서 stock Attribute를 추가하여 특정 매장에 어떤 제품이 얼마나 있는지에 대한 정보를 확인할 수 있다. 추가로, store Entity는 order Entity와 Relation를 형성하여 특정 매장에서 발생한 주문들에 대한 정보를 접근할 수 있게 된다. 이 Entity를 통해 매장에 대한 정보를 입력, 수정, 삭제, 조회할 수 있게 된다.

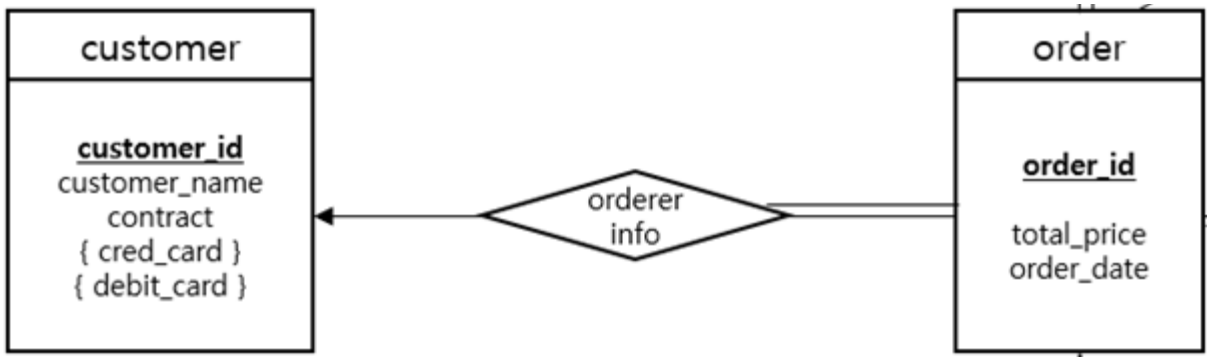
(7) Package



프로젝트의 요구사항중 Gateway Pc와 Sony monitor, HP printer가 한 패키지로 묶일 수 있어야 한다고 했으므로, 물건의 묶음인 패키지에 대한 정보를 저장하는 Entity인 package Entity를 설정했다. 각 패키지마다 다르게 붙는 번호인 package_id, 패키지의 이름 정보를 저장하는 package_name, 그리고 보통 묶음상품은 개별 구매와 달리 추가 할인이 들어가므로 패키지의 가격 정보를 저장하는 package_price 까지 하여 3가지 Attribute로 구성되어 있다. package_id 를 통해 각 패키지를 서로 구별할 수 있으므로 package_id 를 package Entity의 primary key로 설정하였다. product와 Relation을 형성하여 특정 패키지에 어떤 제품들이 들어있는지에 대한 정보를 확인할 수 있도록 한다. 이 Entity를 통해 패키지에 대한 정보를 입력, 수정, 삭제, 조회할 수 있게 된다.

3. E-R Model Relation

(1) Customer – Order Relation

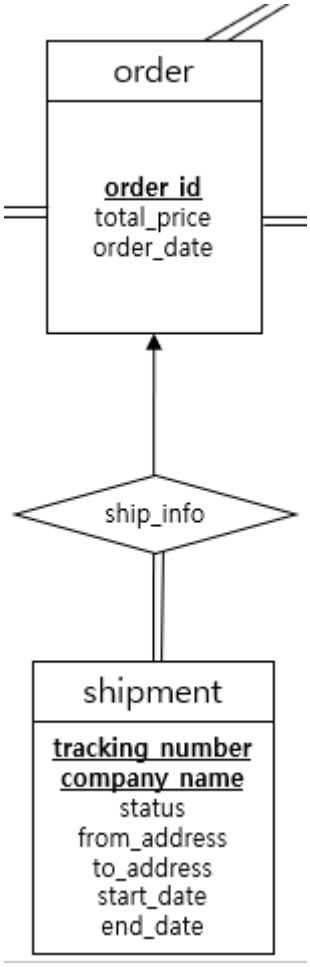


customer Entity와 order Entity 사이에 orderer_info Relation을 형성한다. 이 Relation은 고객이 물건을 살 때 발생하는 주문을 의미한다. 한명의 고객이 여러 번 구매, 즉 여러 번의 주문을 할 수 있는 반면, 하나의 주문은 반드시 한명의 고객을 가지고, 여러 고객을 가지는 것은 불가능하다. 따라서 customer – order 간의 Relation은 one-to-many Relation을 형성한다. 고객이 주문을 한번도 하지 않을 수 있다. 예를들어, 온라인 쇼핑몰에 접속해 회원가입한 고객의 경우 첫 주문 전까지는 아무런 주문 정보가 없을 수 있다. 하지만, 주문은 고객, 즉 주문자가 없을 수 없다. 반드시 주문을 한 고객이 있어야 한다. 따라서 customer는 partial participation 이지만, order는 total

participation 이다.

정리하면, order는 total participation해야 하며, customer와 order 사이의 mapping cardinality는 one-to-many 로 정의할 수 있다.

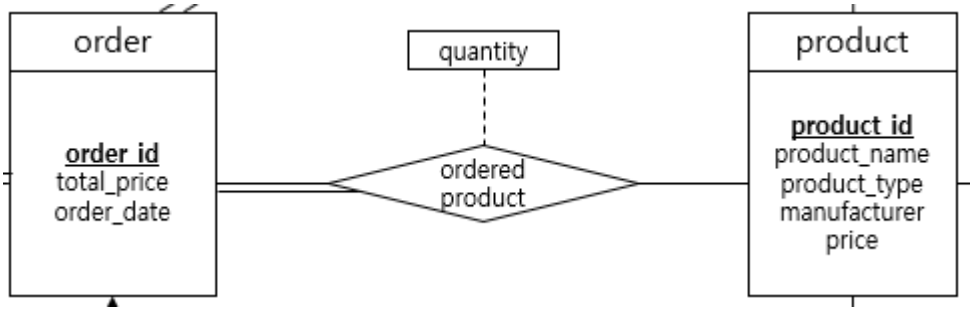
(2) Order – Shipment Relation



order Entity와 shipment Entity 사이에 ship_info Relation을 형성한다. 이 Relation은 주문의 배송 정보를 의미한다. 주문을 한번 하면, 그 주문에 대응하는 배송은 한번 이상이 될 수 있다. 만약 배송중 이상이 있어 재배송이 이루어 질 경우 한 주문에 두번 이상의 배송이 있을 수 있다. 하지만, 한번의 배송에 여러 주문이 있을 수는 없다. 만약 오프라인 매장에서 구매한다면, 손님이 직접 제품을 구매하므로 배송정보가 없을 수 있다. 따라서 order – shipment Relation은 one-to-many Relation을 형성한다. 모든 배송은 주문이 있어야 하므로, shipment는 total participation이다. 반면 주문에 배송이 없을수 있으므로 order는 partial participation이다.

정리하면, shipment 는 total participation해야 하며, order와 shipment 사이의 mapping cardinality 는 one-to-many 로 정의할 수 있다.

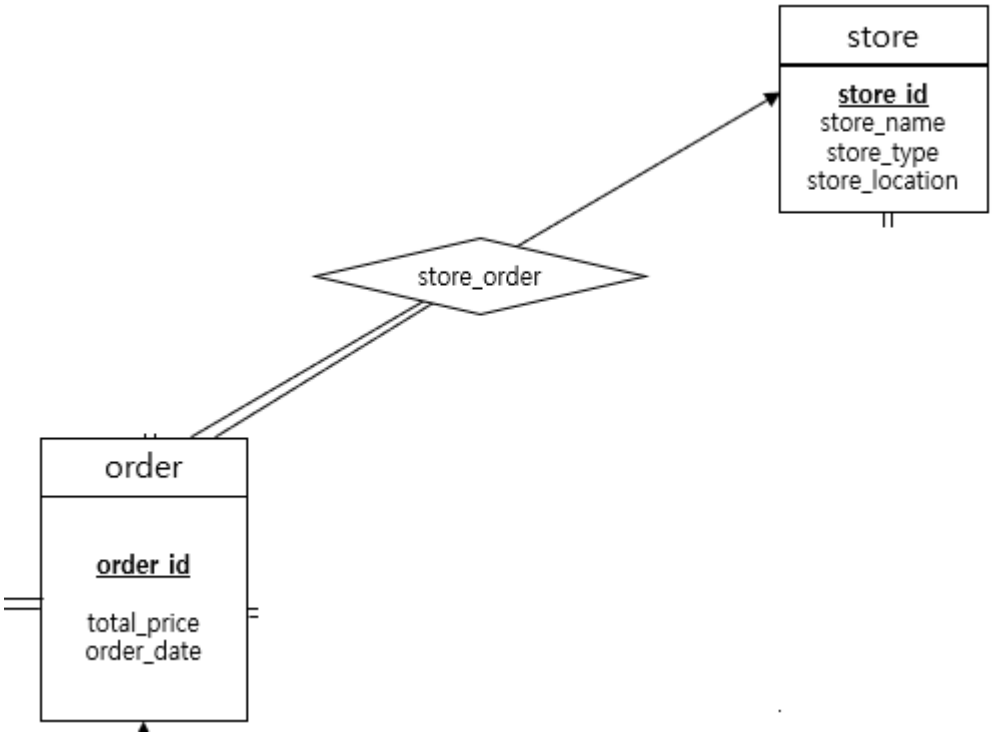
(3) Order – Product Relation



order Entity와 product Entity 사이에 ordered_product Relation을 형성한다. 이 Relation은 주문의 내용물, 즉 어떤 물건을 주문했는지에 대한 정보를 의미한다. 이 Relation에는 extra Attribute 인 quantity Attribute가 추가되어 주문에 포함되는 물건의 수에 대한 정보가 추가된다. 주문을 한번 하면, 여러 종류의 물건을 한번에 주문할 수 있다. 특정 물건이 여러 주문에 포함되어 있을 수 있다. 따라서, order – product Relation은 many-to-many Relation이다. 물건이 전혀 팔리지 않아 특정 물건에 대한 주문이 아예 없을 수 있지만, 주문을 했으면 반드시 하나 이상의 물건을 포함해야 한다. 따라서 product는 partial participation이지만, order는 total participation이다.

정리하면, order는 total participation해야 하며, order와 product 사이의 mapping cardinality는 many-to-many 로 정의할 수 있다.

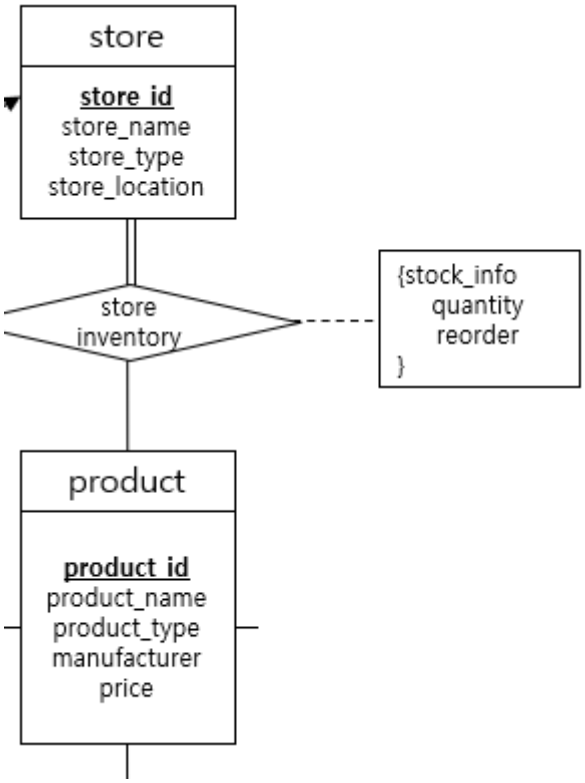
(4) Order – Store Relation



order Entity와 store Entity 사이에 store_order Relation을 형성한다. 이 Relation은 어떤 상점에서 어떤 주문이 이루어졌는지에 대한 정보를 의미한다. 특정 주문이 발생한 상점은 최대 1개로, 한 주문이 여러 상점에서 이루어질 수는 없다. 하지만 한 상점에서 여러 주문이 이루어질 수는 있다. 따라서, order – store Relation은 many-to-one Relation이다. 주문은 반드시 하나의 매장에서 이루어진다. 하지만, 특정 매장에서 아무런 주문도 발생하지 않을 수 있다. 따라서, order는 total participation, store는 partial participation이다.

정리하면, order는 total participation해야 하며, order와 store 사이의 mapping cardinality는 many-to-one 으로 정의할 수 있다.

(5) Product – Store Relation

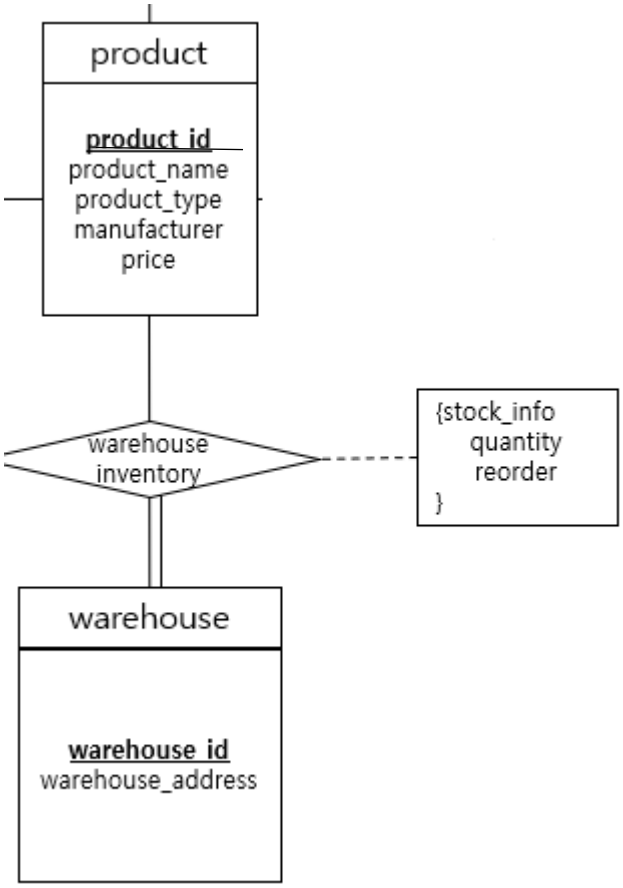


product Entity와 store Entity 사이의 store_inventory Relation을 형성한다. 이는 매장에서 판매하는 제품 종류를 의미한다. 이 Relation에는 extra Attribute인 stock_info Attribute를 추가한다. stock_info attribute는 quantity와 reorder 두가지 subpart를 갖는 composite attribute로 설정했다. 이 attribute의 quantity는 상점에 있는 특정 제품의 수량과, 재주문 상태를 나타낸다. 프로젝트 설명에 만약 제품 재고가 부족할경우 제조사에 재주문을 넣고, 제품이 도착하면 reorder로 제품이 보충됐다고 데이터베이스에 update하라는 내용이 있어, 해당 속성을 추가했다. 이 속성을 통해 현재 어떤 상점에 어떤 제품이 부족하여 재주문중인지, 혹은 재주문이 끝났는지 확인할 수 있다.

매장 하나에 여러 종류의 제품을 팔 수 있고, 제품 하나는 여러 매장에 존재할 수 있다. 따라서 product - store Relation은 many-to-many Relation이다. 매장에서 판매하는 제품은 반드시 하나 이상 존재해야 하지만, 특정 제품이 아무데서도 판매되지 않을 수 있고, 아니면 온라인 매장에서만 판매될 수 있다. 따라서, store는 total participation 하지만, product는 partial participation한다.

정리하면, store는 total participation해야 하며, product 와 store 사이의 mapping cardinality는 many-to-many 로 정의할 수 있다.

(6) Product – Warehouse Relation

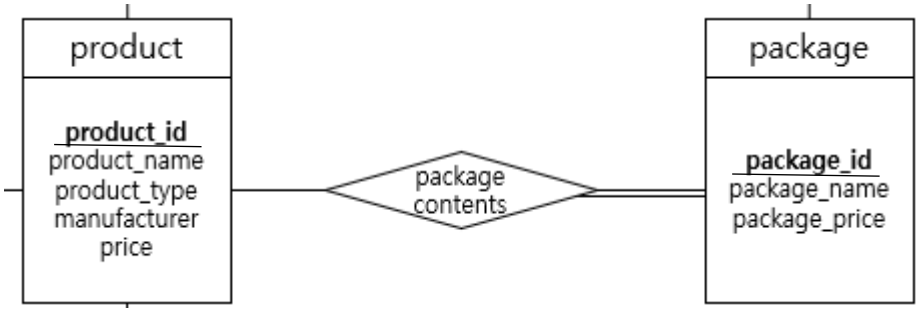


product Entity와 warehouse Entity 사이에 warehouse_inventory Relation을 형성한다. 이는 창고에 보관하는 제품의 종류를 의미한다. 이 Relation에 역시 창고에 보관되는 제품의 수량과, 해당 제품의 재주문 상태를 나타내는 extra Attribute이며 composite attribute인 stock_info Attribute가 추가된다.

하나의 제품이 여러 창고에 보관될 수 있고, 한 창고에 여러 제품을 보관할 수 있으므로 product – warehouse Relation은 many-to-many Relation이다. 특정 제품은 아무런 창고에 보관되지 않을 수 있지만, 특정 창고가 아무 제품도 보관하지 않을수는 없다. 따라서, product는 partial participation, warehouse는 total participation한다.

정리하면, warehouse는 total participation 해야하며, product와 warehouse 사이의 mapping cardinality는 many-to-many 로 정의할 수 있다.

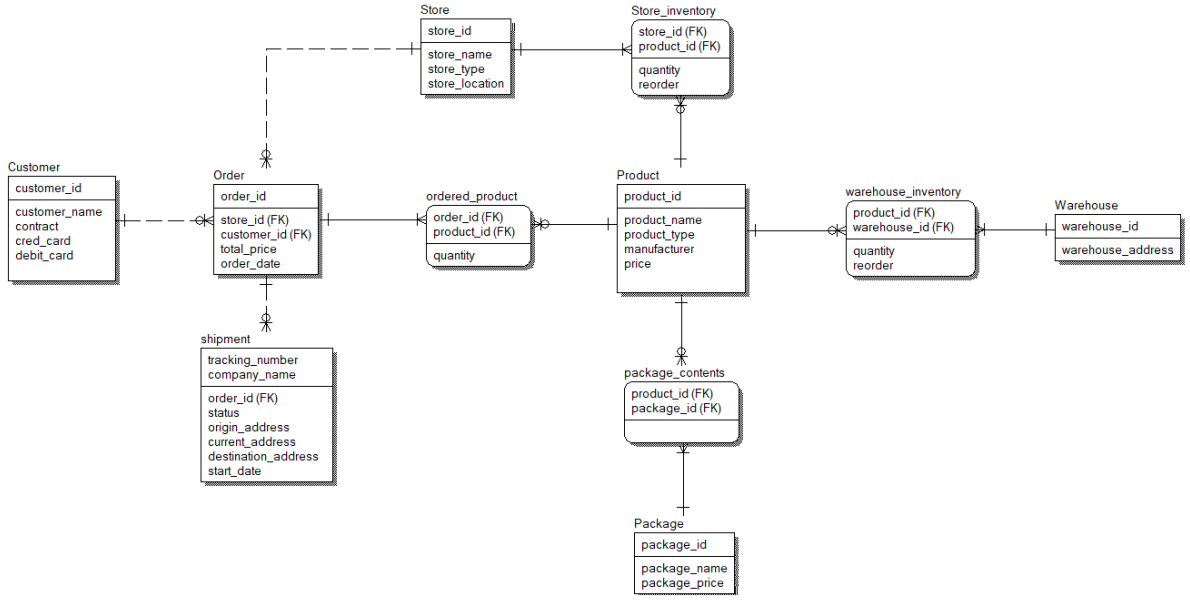
(7) Product – Package Relation



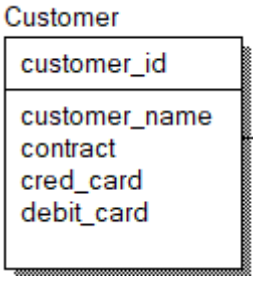
product Entity와 package Entity 사이에 package_contents Relation을 형성한다. 이 Relation은 특정 패키지로 묶인 제품을 의미한다. 한 제품이 여러 패키지에 포함될 수 있고, 반대로 한 패키지에 여러 제품이 포함될 수 있다. 따라서, 둘 사이의 관계는 many-to-many Relation을 형성한다. 특정 제품은 아무 패키지에 포함되지 않고 개별판매만 될 수 있지만, 패키지는 반드시 하나 이상의 물건을 포함해야 한다. 따라서, product는 partial participation, package는 total participation 해야한다.

정리하면, package는 total participation 해야하며, product와 package 사이의 mapping cardinality는 many-to-many 로 정의할 수 있다.

4. Relational Schema Diagram



먼저 기존 ER model의 Entity들을 살펴본다.



customer Entity의 primary key는 customer_id 이다.

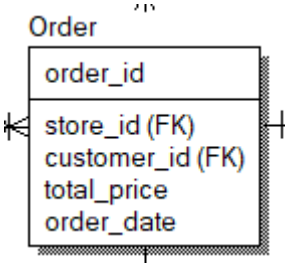
customer_id는 Integer type의 데이터를 가지며 Null이 될 수 없다.

customer_name은 긴 이름을 고려하여, CHAR(50) type의 데이터를 가질 수 있어 최대 50자까지의 이름을 지원한다. 또한, Null이 될 수 없다.

contract는 계약 여부만 판단하므로, BOOLEAN type의 데이터를 가져 계약을 했으면 true, 아니면 false 값을 가진다. 역시 Null이 될 수 없다.

cred_card는 신용카드로, 0000-0000-0000-0000 형태의 문자열로 카드정보를 저장하므로 CHAR(19 type의 데이터를 가지며, 카드정보가 등록되지 않은 고객이 있을 수 있으니 Null을 허용한다.

debit_card 역시 cred_card와 동일한 이유로 CHAR(19) type 데이터를 저장하며 Null을 허용한다.



order Entity의 primary key는 order_id이다.

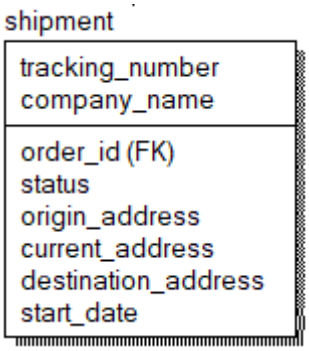
order_id는 Integer type의 데이터를 가지며 Null이 될 수 없다.

store_id는 store Entity와의 Relation으로부터 비롯된 foreign key이다. 후에 설명한다.

customer_id는 customer Entity와의 Relation으로부터 비롯된 foreign key이다.

total_price는 Integer type의 데이터를 가지며 Null이 될 수 없다.

order_date는 주문 날짜를 나타내므로, erwin에서 지원하는 data type중 DATETIME DAY TO SECOND 를 사용하여 주문 날짜를 초 단위로 저장한다. Null이 될 수 없다.



shipment의 primary key는 tracking_number, company_name 두개이다.

tracking_number는 Integer type 데이터를 가지며 Null이 될 수 없고, company_name은 CHAR(50) type의 데이터를 가져 최대 50자까지의 회사 이름을 저장할 수 있다. Null이 될 수 없다.

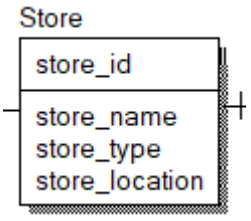
order_id는 order Entity와의 관계에서 비롯된 foreign key이다. 후에 설명한다.

status는 CHAR(10) type으로 배송중, 배송준비중, 배송실패 등의 정보가 들어가므로 최대 10자로 정했으며, Null이 될 수 없다.

origin_address, current_address, destination_address 모두 주소를 나타내는 Attribute이므로 CHAR(100) type의 데이터를 저장하여 주소를 나타낸다. 모두 Null이 될 수 없다.

start_date는 배송이 시작된 날짜를 의미하므로 DATETIME DAY TO SECOND type의 데이터를 저장하며, Null이 될 수 없다.

end_date 는 배송이 끝난 날짜를 의미하여 동일하게 DATETIME DAY TO SECOND type의 데이터를 저장하지만, 배송이 끝나기 전에는 값이 결정되지 않으므로 Null을 허용한다.



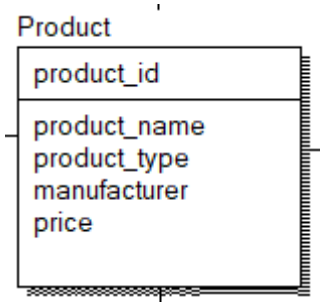
store Entity의 primary key는 store_id 이다.

store_id는 Integer type 데이터를 가지며 Null이 될 수 없다.

store_name은 CHAR(50) type의 데이터를 가져 최대 50자까지의 상점이름을 저장한다. Null이 될 수 없다.

store_type은 online / offline 으로 구분하므로 CHAR(7) type의 데이터를 가진다. Null이 될 수 없다.

store_location은 주소를 나타내므로 CHAR(100) type 데이터를 저장한다. Null이 될 수 없다.



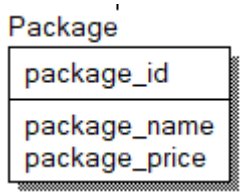
product의 primary key는 product_id이다.

product_id는 Integer type 데이터를 가지며 Null이 될 수 없다.

product_type은 물건의 종류로, CHAR(20) type data를 가지며 Null이 될 수 없다.

manufacturer는 물건의 제조사로, CHAR(50) type data를 가지며 Null이 될 수 없다.

price는 물건의 가격으로 Integer type 데이터를 가지며 Null이 될 수 없다.

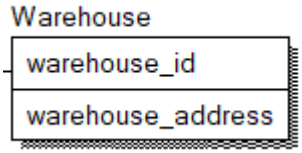


package Entity의 primary key는 package_id이다.

package_id는 Integer type 데이터를 가지며 Null이 될 수 없다.

package_name은 CHAR(50) type 데이터를 가지며 Null이 될 수 없다.

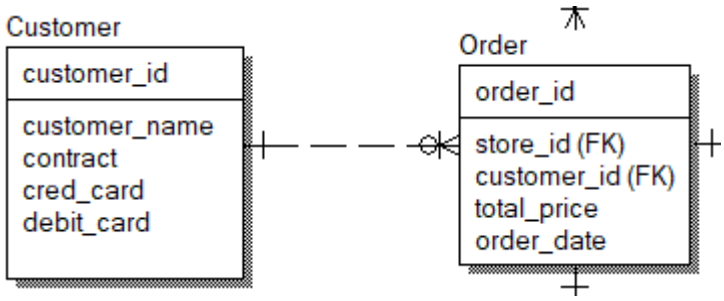
package_price는 Integer type 데이터를 가지며, Null이 될 수 없다.



warehouse Entity의 primary key는 warehouse_id이다.

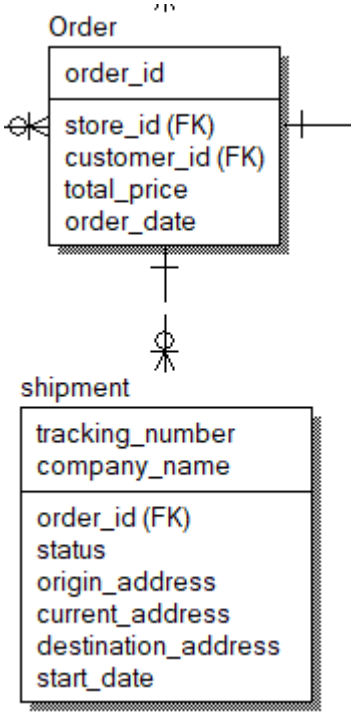
warehouse_id는 Integer type 데이터를 가지며 Null이 될 수 없다.

warehouse_address는 주소를 나타내므로 CHAR(100) type 데이터를 갖고, Null이 될수 없다.



customer – order 사이 관계는 order는 total participation해야 하며, customer와 order 사이의 mapping cardinality는 one-to-many 이다.

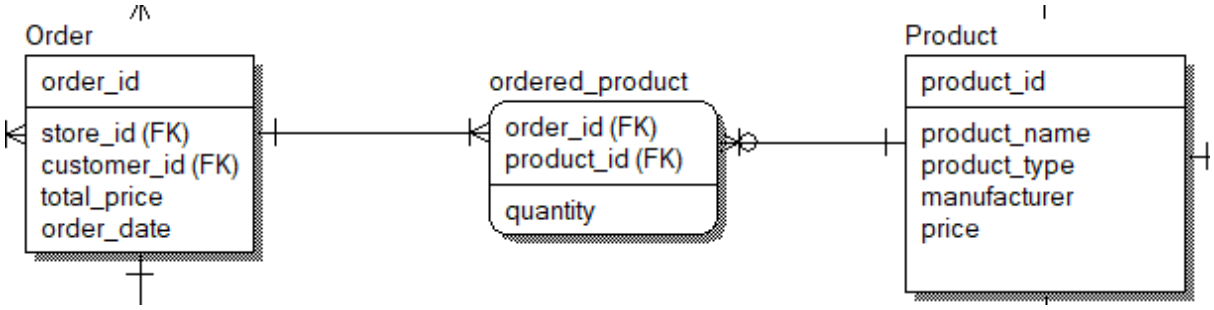
다시말해, customer는 order를 하나도 하지 않을 수 있고, 하나만 할 수 있으며, 여러 order를 할 수도 있다. 따라서, customer – order 사이의 relationship은 one-to-zero-one-or-more 로 설정한다. schema reduction rule에 따라, order는 customer의 primary key를 foreign key로 갖게 된다. 이때 customer_id 는 order의 primary key로 불필요하므로, 둘 사이의 relationship은 non-identifying relationship이다.



order – shipment 사이 관계는 shipment가 total participation해야 하며, order와 shipment 사이의 mapping cardinality는 one-to-many 이다. 다시말해, shipment는 하나의 order를 갖고, order는 shipment를 갖지 않을수도, 하나 이상을 가질수도 있다.

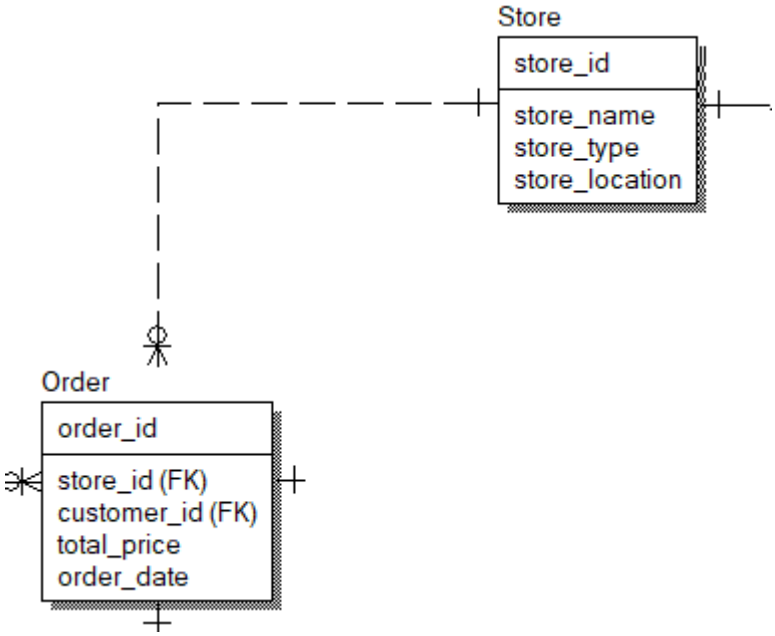
즉, order - shipment의 Relation은 one-to-zero-one-or-more 관계로 설정한다. 이에 따라 shipment는 order의 primary key를 foreign key로 갖는데, 그것이 order_id이다. 이때 order_id는

shipment의 primary_key로 불필요하므로 둘 사이의 relationship은 non-identifying relationship이다.



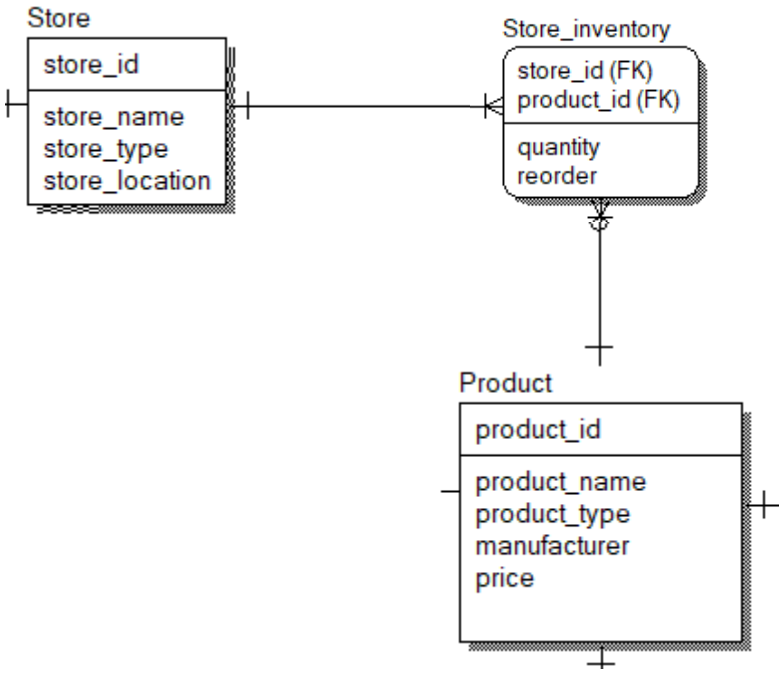
order와 product 사이 관계는 order는 total participation해야 하며, order와 product 사이의 mapping cardinality는 many-to-many 이다. many-to-many relationship이므로, 새로운 schema ordered_product를 생성한다. 이 schema는 order로부터 order_id를 , product 로부터 product_id를 primary key이자 foreign key로 받는다. 추가로, extra attribute인 quantity를 갖는다. 따라서, order와 ordered_product, ordered_product와 product 사이의 관계는 모두 identifying relationship이다.

한번의 주문은 여러 제품을 포함할 수 있고, 아무 제품도 없을 수는 없으므로, order와 ordered_product 사이의 relationship은 one-to-one-or-more relationship이 성립한다. 한 제품이 여러 주문에 들어있을 수 있고, 아무 주문에도 안들어있어도 괜찮으므로 product와 ordered_product 사이의 relationship은 one-to-zero-one-or-many relationship이 성립한다.



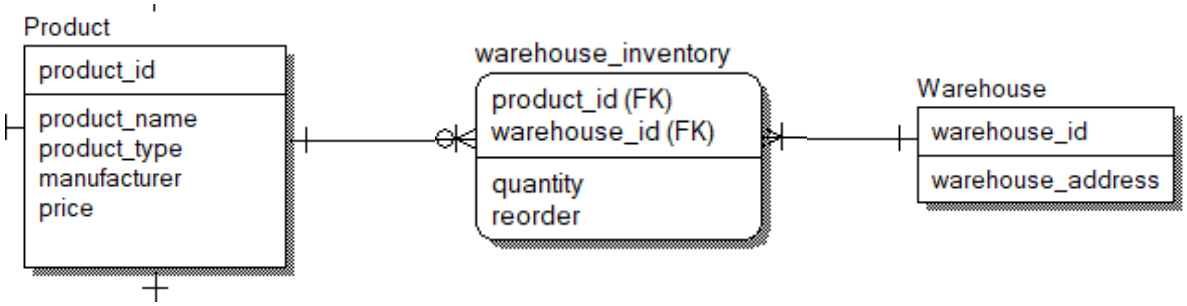
order와 store 사이의 관계는 order는 total participation해야 하며, order와 store 사이의 mapping cardinality는 many-to-one 이다. 즉, 한 매장이 많은 주문을 가질수 있지만, 한 주문이 많은 매장을 가질 순 없다. 한 매장은 주문이 없어도 되므로, store와 order 사이의 관계는 one-to-zero-

one-or-more 관계를 이루게 된다. 이에 따라 order는 store의 primary key인 store_id를 foreign key로 받게 되는데, order의 primary key로 필요하지 않으므로 둘 사이의 관계는 non-identifying relationship이다.

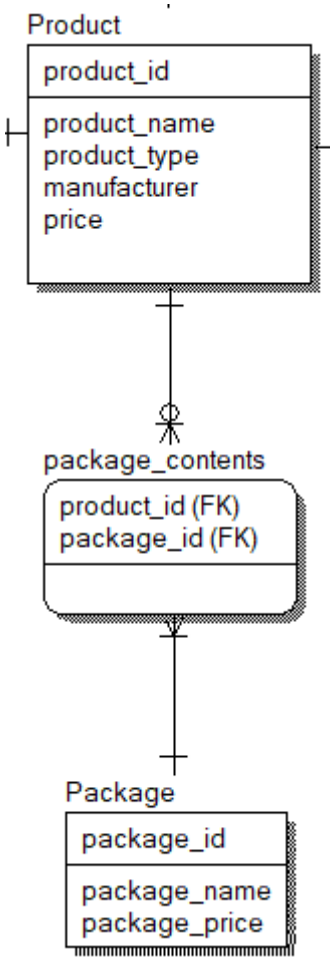


product와 store 사이의 관계는, store는 total participation해야 하며, product 와 store 사이의 mapping cardinality는 many-to-many 이다. 이에 따라 store와 product 사이에 별도의 schema를 생성한다. 이를 store_inventory 라고 한다. 이 schema는 store 로부터 store_id를, product로부터 product_id를 받고, 추가적인 attribute로 quantity와 reorder를 갖는다. 그 이유는 composite attribute의 경우 각 component attribute를 분리하여 추가하기 때문에, 기존의 stock_info를 위와 같은 두 attribute로 분리하여 추가한다. quantity는 수량을 나타내므로 Integer type의 데이터를 갖고, reorder는 재주문의 상태를 나타내므로 CHAR(20) type의 데이터를 저장한다. 둘 모두 Null이 될 수 없다. foreign key가 primary key로 기능하고 있으므로, store와 store_inventory, product와 store_inventory 관계 둘 모두 identifying relationship이다.

한 상점에 여러 제품이 있을 수 있지만, 상점에서 아무 제품도 팔지 않을 수는 없으므로 store와 store_inventory 사이의 관계는 one-to-one-or-more relationship이 성립한다. 한 제품이 여러 상점에 있을 수 있고, 특정 제품은 아무 상점에서도 팔지 않을 수 있다. 따라서, product와 store_inventory 사이의 relationship은 one-to-zero-one-or-more relationship이 성립한다.



product와 warehouse 사이의 관계는, warehouse는 total participation 해야하며, product와 warehouse 사이의 mapping cardinality는 many-to-many 이다. many-to-many relationship이므로 사이에 별도의 schema인 warehouse_inventory를 생성한다. 이는 product_id와 warehouse_id를 foreign key이면서 primary key로 갖고, 앞서 설명한 store-product 관계와 동일하게 quantity와 reorder를 Attribute로 갖는다. 이유 역시 위와 동일하다. foreign key가 primary key로 기능하므로, product와 warehouse_inventory , warehouse와 warehouse_inventory 두 relationship 모두 identifying relationship이다. 한 제품이 여러 창고에 있을 수 있고, 아무 창고에 없을수도 있다. 따라서, product와 warehouse_inventory는 one-to-zero-one-or-many relationship이 성립한다. 하지만, 한 창고에 여러 제품이 있을 수는 있지만, 창고에 아무 제품도 보관되지 않을 수는 없으므로 warehouse와 warehouse_inventory 사이의 relationship은 one-to-one-or-more relationship이 성립한다.



product와 package 사이의 관계는 package는 total participation 해야하며, product와 package 사이의 mapping cardinality는 many-to-many 이다. 다른 경우들과 동일하게 별도의 schema를 형성하고, product_id와 package_id 두가지를 foreign key이자 primary key로 가지므로 두 relationship 모두 identifying relationship이며, 한 제품이 여러 패키지에 들어있을수도, 아무 패키지에 없을수도 있으므로 one-to-zero-one-or-more relationship, 한 패키지에 여러 제품이 있을순 있지만 아무 제품이 없을 수는 없으므로 one-to-one-or-more relationship이 성립한다.