**데이터베이스시스템**

Project1 보고서

정성원 교수님

01반

20181297

수학과

조다민

1. **프로젝트 개요**

이번 프로젝트의 목표는 온라인 매장과 여러 개의 오프라인 매장을 동시에 가진 전자상가에 대한 정보를 저장하고 또 관리할 수 있는 데이터베이스를 구축하는 것이다. 이 프로젝트를 통해 판매되는 상품, 매장, 고객 정보, 재고 상태, 고객 주문 내역 등의 내용을 관리할 수 있는 데이터베이스에 대한 기초적인 정보를 담은 ER-diagram, Relation schema, 그리고 몇 가지 예시 SQL문을 작성하였다.

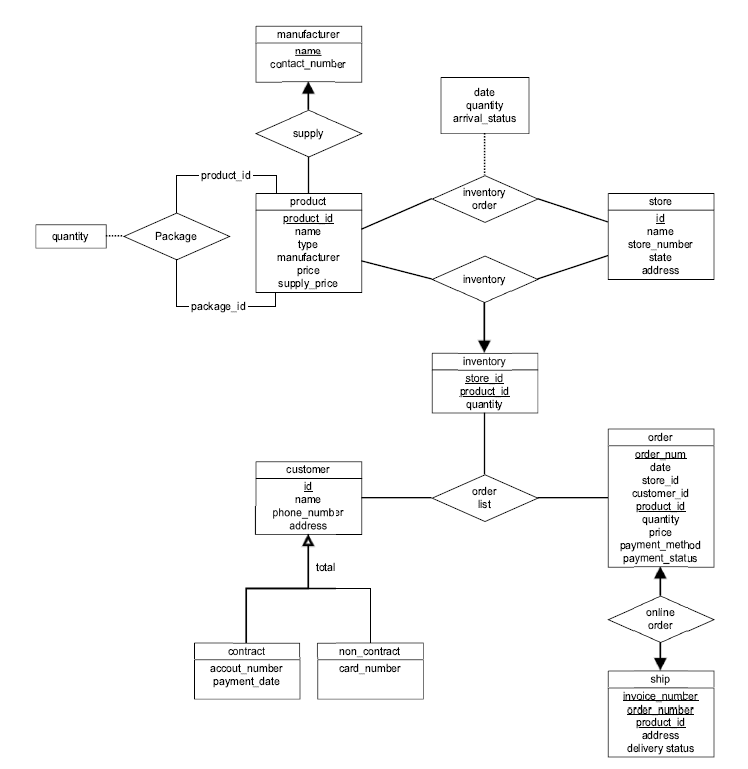
1. **ER-diagram**

Figure : ER-diagram 작성 결과.

**2.1 Entity set에 대한 설명**

**진한 글씨로 표시한 attribute는 primary key임을 뜻한다.**

* + 1. product

회사에서 다루는 모든 제품에 대한 정보를 담고 있다. 속성으로는 **product\_id**, type, manufacturer, price, supply\_price 가 있고 각각은 제품 고유 id, 분류(모니터, 케이블 등), 제조사, 판매가, 원가 정보를 저장한다.

* + 1. Manufacturer

Product를 생산하는 제조사에 대한 정보를 담고 있다. 이 entity는 회사에서 제조사에 재고 구매 요청할 때 필요한 회사의 이름(**name**)과 회사의 번호(contact\_number)를 저장한다.

* + 1. Store

온, 오프라인 매장에 대한 정보를 담고 있다. attribute로는 매장 고유 id(**id**), 매장명(name), 매장 연락처(store\_number), 매장이 위치한 주 또는 도시(state), 매장 주소(address) 정보를 저장한다. 온라인 매장의 경우 state와 address값으로 warehouse의 정보를 가지고 있다.

* + 1. Inventory

Inventory의 경우 각 매장 별 재고를 저장하고 있다. 온라인 매장에 대한 재고는 전체 오프라인 매장에 필요한 재고를 비축해두는 warehouse의 재고와 동일하게 보고 매장 id(**store\_id**), 제품 id(**product\_id**), 수량(quantity)를 저장한다.

* + 1. Customer

회사를 이용하는 고객들에 대한 정보를 담고 있다. 기본적으로 고객 id(**id**), 이름(name), 핸드폰 번호(phone\_number), 주소(address)에 대한 정보를 담고 있으며, 하위 속성에 따라 contract와 non-contract로 구분한다. Contract 고객의 경우에는 계약을 통해 한달 동안의 구매에 대한 결제를 지정 계좌를 통해 일정 일자에 수행하기 때문에 지정계좌(account\_number)와 결제 일자(payment\_date)를 저장하고 있다. Non-contract 고객의 경우에는 online 결제와 offline 결제의 정보가 다른데, online 구매의 경우 추후 빠른 정보 입력을 위해 카드 번호(card\_number)가 저장되며 offline 구매의 경우 다른 정보 저장은 없다.

* + 1. Order

Order에서는 고객의 구매와 관련된 정보를 저장하고 있으며, 온-오프라인 모두에 해당하는 정보들을 담고 있다. 저장하는 정보로는 주문번호(**order\_num**), 주문 일자(date), 주문매장(store\_id), 주문 고객(customer\_id), 상품 정보(**product\_id**), 수량(quantity), 가격(price), 지불 수단과 지불 상태(payment\_method, payment\_status)가 있다. 여기서 주문번호 하나만으로는 tuple을 구분할 수 없다. 한번에 여러 개의 상품을 주문했을 때가 있기 때문에 (order\_num, product\_id)를 primary key로 한다.

* + 1. Ship

Ship의 경우 배송이 필요한 online 구매에 대해 배송 정보를 다룬다. 배송 송장번호(**invoice\_number**), 주문번호(**order\_number**), 제품번호(**product\_id**), 주소(address), 배송 상태(delivery status)를 저장하고 있다. 제품 여러 개가 하나의 송장번호로 배송될 수 있어서 송장번호, 주문번호, 제품번호 모두를 primary key로 한다.

**2.2 Relation set에 대한 설명**

1. Package

여기서는 type 중 package 상품에 대한 정보는 추가적으로 relation을 이용해 저장하고 있다. 패키지 상품에 대한 정보를 저장하는 relation이다. package\_id로 원래 패키지 상품의 product\_id를 가져와서 패키지에 포함되는 product\_id와 개수(quantity)를 연결하는 다대다 관계이다.

1. Supply

Product와 manufacturer 사이를 연결한다. 이를 통해 재고가 부족하거나 제품에 문제가 있을 때 manufacturer에 바로 연락할 수 있는 연락망을 저장할 수 있으며 하나의 회사가 여러 개의 제품을 만들고 납품하므로 회사-제품 관계는 일대다 관계로 볼 수 있다.

1. Inventory

매장과 제품 정보와 재고상태를 inventory relation으로 연결하는 ternary관계이다. 재고상태로 여러 매장과 제품 정보가 전달되므로 재고상태는 one 나머지는 many로 볼 수 있다.

1. Inventory order

재고가 부족한 경우 제조사로 재고 주문을 넣는 것을 나타내는 relation으로 매장(id)와 제품(id) 정보 외에 추가적으로 주문일자(date), 수량(quantity), 도착여부(arrival\_status) 정보를 알고 있어야한다.

1. Order list

재고와 고객정보, 주문을 연결하는 ternary relation이다. 다대다 관계를 가지고 있다.

1. Online order

Order와 배송정보(ship)을 연결하는 relation이다. Order\_number와 product\_id를 통해 연결할 수 있고 일대일 관계를 가진다.

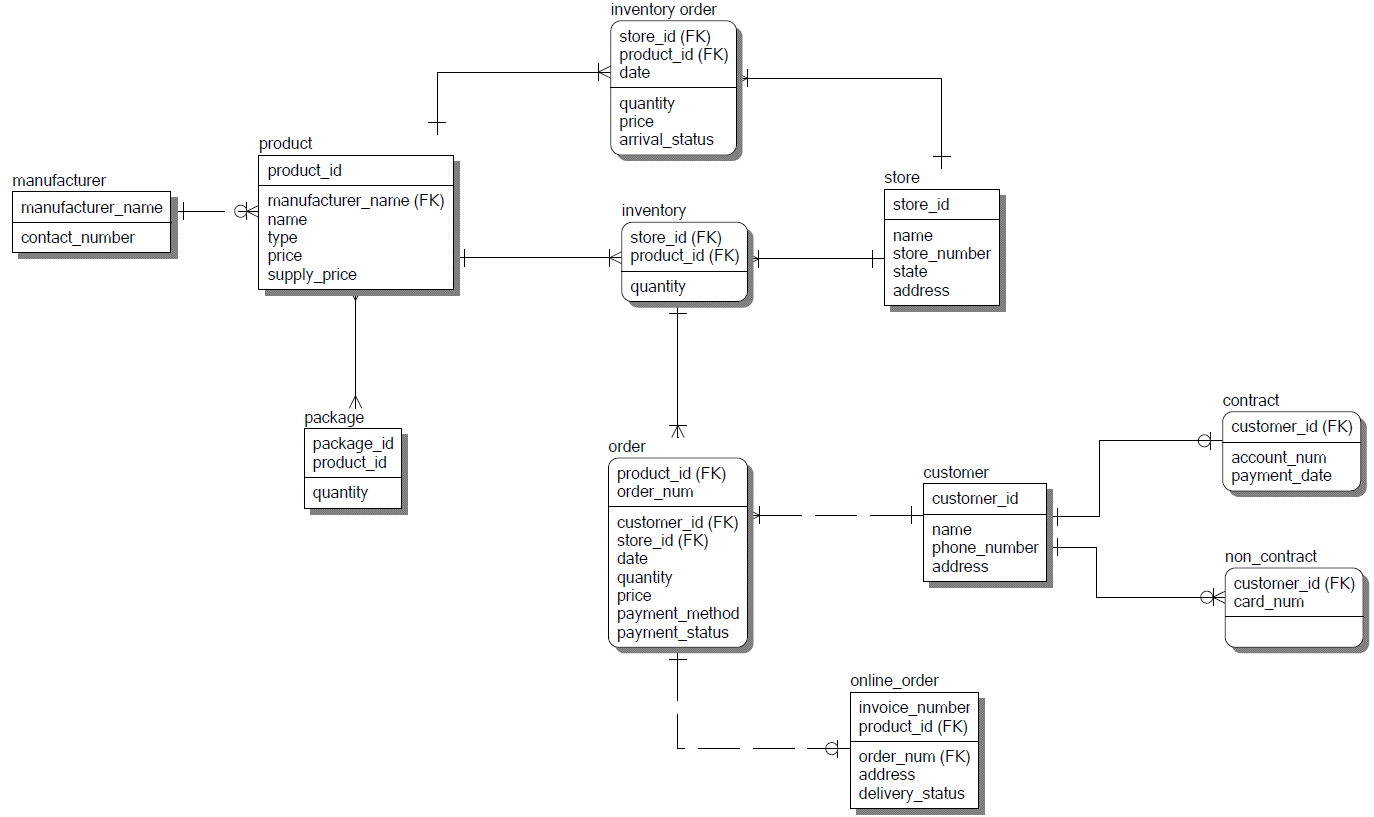
1. ** Relation Schema**

Figure : ER diagram을 바탕으로 작성된 relation schema

* 1. Entity

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entity | attribute | Key | Domain | discription |
| product | product\_id | PK | String | 제품 고유 ID |
| manufacturer\_name | FK | String | 제조사명 |
| type |  | String | 분류 |
| price |  | Number | 판매가 |
| supply\_price |  | Number | 원가 |
| manufacturer | manufacturer\_name | PK | String | 제조사명 |
| contact\_number |  | Number | 연락처 |
| package | package\_id | PK | String | 패키지 ID |
| product\_id | PK | String | 패키지에 포함된 상품 ID |
| quantity |  | Number | 상품 수량 |
| store | store\_id | PK | String | 매장 ID |
| name |  | String | 매장명 |
| store\_number |  | Number | 매장번호 |
| state |  | String | 매장이 위치한 주 |
| address |  | String | 매장 주소 |
| inventory | store\_id | PK, FK | String | 매장 ID |
| product\_id | PK, FK | String | 제품 ID |
| quantity |  | Number | 재고 수량 |
| inventory order | store\_id | PK, FK | String | 매장ID |
| product\_id | PK, FK | String | 제품 ID |
| date | PK | Datetime | 주문 일자 |
| quantity |  | Number | 주문 수량 |
| price |  | Number | 결제가 |
| arrival\_status |  | String | 배송상태 |
| Customer | customer\_id | PK | String | 고객 ID |
| name |  | String | 고객명 |
| phone\_number |  | Number | 고객 번호 |
| address |  | String | 고객 주소 |
| contract | customer\_id | PK, FK | String | 고객 ID |
| account\_num |  | Number | 계좌번호 |
| payment\_date |  | Number | 결제일 |
| non-contract | customer\_id | PK, FK | String | 고객 ID |
| card\_num | PK | Number | 결제 카드번호 |
| order | order\_num | PK | Number | 주문번호 |
| product\_id | PK, FK | String | 상품 ID |
| customer\_id |  | String | 고객 ID |
| date |  | Datetime | 주문일자 |
| store\_id | FK | String | 매장 ID |
| quantity |  | Number | 수량 |
| price |  | Number | 가격 |
| payment\_method |  | String | 지불 방법 |
| payment\_status |  | String | 지불 상태 |
| online\_order | invoice\_number | PK | Number | 송장번호 |
| order\_num | FK | Number | 주문번호 |
| product\_id | FK | String | 상품 ID |
| address |  | String | 주소 |
| delivery\_stauts |  | String | 배송상태 |

* 1. Relationship
     1. Product-manufacturer

제품들이 제조사로 연결된다. 이때, 패키지 상품의 경우 제조사를 분명히 할 수 없으므로 표기하지 않는다. 따라서 product.manufacturer\_name은 null이 될 수 있다. 또한 여러 개의 제품이 하나의 회사로 연결되므로 product-manufacturer 관계는 many(zero, one-or more)-to-one이다.

* + 1. Product-package

Product-package 관계는 many-to-many 관계로 정의할 수 있다. Package가 안에 포함하고 있는 상품의 개수만큼 다대다 관계가 생긴다. 물론 package 상품이 아닌 경우 package와의 관계가 없을 수 있다.

* + 1. Product-inventory(inventory\_order), store-inventory(inventory\_order)

Product-inventory, store-inventory 관계는 모두 one-to-many 관계로 정의할 수 있다. 하나의 product에 store의 개수만큼 inventory가 생길 수 있고, 하나의 store에 product 개수만큼 inventory가 생길 수 있다. 또한 store\_id와 product\_id는 모두 product와 store의 내용을 참조해야 한다.

* + 1. Inventory-order

Inventory-order 관계는 여러 개의 주문이 하나의 inventory로 연결될 수 있으므로 one-to-many 관계로 하고, 모든 order는 inventory와 무조건 연결되어야 한다.

* + 1. Order-customer

Order-customer 관계에서 여러 개의 주문이 하나의 고객과 연결되므로 many-to-one 관계로 정의할 수 있다. 또한 모든 order는 customer 정보를 담고 있어야 한다.

* + 1. Order-online\_order

Order-online\_order 관계에서 order가 online인 경우만 관계가 생성되고, 송장번호와 product로 구분되는 online-order에 대하여 물건이 교환 등의 이유로 재배송 되는 경우가 있을 수 있기 때문에 하나의 order에 대해 여러 개의 online\_order가 연결될 수 있어 one-to-many관계로 정의할 수 있다.

* + 1. Customer-contract

계약을 통해 주문을 하는 고객들은 하나의 account\_number 정보를 가지고 있어야하므로 one-to-one 관계를 가지며 계약이 없는 경우는 contract와 관계가 없을 수 있다.

* + 1. Customer-non\_contract

계약이 없는 online 주문 고객들의 카드 정보를 저장하는데, 이때 카드 정보 여러개가 하나의 고객과 연결될 수 있으므로 one-to-many의 관계를 가진다.

1. **예시 sql문**
   1. 송장번호가 123456인 배송 중 파손 상품 고객의 번호를 찾기

select phone\_number

from customer natural join (select \* from order, online\_order

where (order.order\_num = online\_order.order\_num)

and (order.product\_id = online\_order.product\_id))

where invoice\_number="123456"

* 1. 파손된 배송 상품에 포함된 상품을 확인

select order.product\_id, order.quantity from order, online\_order

where (order.order\_num = online\_order.order\_num)

and (order.product\_id = online\_order.product\_id)

and (invoice\_number="123456")

* 1. 새로운 배송정보를 만들어 재발송

with destroyed as

(select order.order\_num as num, order.product\_id as p, address as a

from order, online\_order

where (order.order\_num = online\_order.order\_num)

and (order.product\_id = online\_order.product\_id)

and (invoice\_number="123456"))

insert into online\_order

("123457", destroyed.num, destroyed.p, destroyed.a, "started")

* 1. 작년(2021)에 가장 많은 금액을 사용한 고객

select customer\_id, name

from customer

where (select customer\_id, sum(price)

from order

where DATEOFYEAR(date) = 2021

group by customer\_id)

* 1. 작년(2021)에 가격적으로 많이 팔린 상품 상위 2개

select product\_id, sum(price)

from order

where DATEOFYEAR(date) = 2021

group by product\_id

order by sum(price) desc

LIMIT 2;

* 1. 작년(2021)에 수량적으로 가장 많이 팔린 상품 상위 2개

select product\_id, sum(quantity)

from order

where DATEOFYEAR(date) = 2021

group by product\_id

order by sum(quantity) desc

LIMIT 2;

* 1. 캘리포니아 지역 전체에서 품절인 상품

select product\_id

from product

where product\_id not in

(select \* from inventory natural join store

where store.state = "California"

and inventory.quantity >0)

F. 고객별 지난달(3월) 구매 금액의 총합

select customer\_id, sum(price)

from order

where payment\_status = "No" and DATEOFMONTH(date)=3

group by customer\_id

* 1. 아직 도착하지 않은 택배들 중 약속된 날짜(3일)안에 도착하지 않은 택배.

select invoice\_number

from (select \* from order, online\_order

where (order.order\_num = online\_order.order\_num)

and (order.product\_id = online\_order.product\_id)

)

where (DATEIFF(dd, date(now()), date(order.date())>3)

and (online\_order.delivery\_status != "complete")