학번 : 20210354

이름 : 김상협

<ER diagram>

텍스트, 도표, 평면도, 평행이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

( Weak entity set 에서 entity 가 두줄로 바뀌는 걸 계속 찾지 못해서 weak entity 에서는 바깥 선을 점선으로 표시했습니다!)

1. 서론

이번 2025 spring cse4100 project 1 에서 편의점을 운영하는 enterprise 에서 database 를 구성하는 것을 가정하고 database의 conceptual design 을 수행하였습니다. 편의점내 여러가지 운영 및 검색을 할 수 있도록 ER 다이어그램을 작성했습니다. 이후 보고서에는 작성한 ER 다이어그램에 대한 엔티티와 Relation 에 대해서 더 자세하게 상술하겠습니다.

1. 비즈니스 규칙

Store : 편의점은 가맹점/직영점으로 나뉘어 있으며 , 각자 고유한 store\_id 를 가짐.

Products : 모든 물건에는 고유한 UPC 가 부여되어 있어서 식별가능하다. 이외에도 물건의 브랜드,가격,사이즈,이름까지 다양한 정보가 저장되어 있다.

Vendors : 상품을 공급하는 업체이다. 고유한 id 가 있다. Vendor 에 대한 내용들이 추가로 저장되어 있다. 한 vendor 가 여러 개의 products 를 제공할 수 있다.

Inventory : 편의점 점포가 가지고 있는 상품들을 나타낸 정보이다. 한 편의점이 여러개의 물건을 가지고 있을 수 있으며, 물건에 대한 정보는 Products entity 를 참고 한다.

Sale: customer 가 구매한 물건에 대한 내용을 저장하고 있는 entity 이다. 물건에 대한 자세한 정보는 sale\_detail entity 를 만들어 어떤 물건을 구매했는지 조회할 수 있도록 하였다.

Sale\_detail : 고객이 구매한 물건에 대한 상세한 정보를 가지고 있습니다.

Customer : 물건을 구매한 고객이다. Customer\_Id 값을 가지고 있으며, membership\_point 를 통해 vip 고객인지 아닌지 판단할 수 있도록 구성하였다. 이때 membership point 가 0인지 아닌지를 통해서 비회원/회원 인지도 구분할 수 있다.

1. Entity

<**Store**> : store\_id(PK) , store\_name , address ,open\_time , close\_time , ownership\_type

(Store\_id)\_ : pk 로 가지고 있어 entity 내 식별 할 수 있도록 하였다.

(ownership\_type) : 해당 매장이 직영점인지 가맹점인지 구분되도록 하였다.

(opentime) , (close\_time) : 24 시간 영업중인지 아닌지는 close\_time-open\_time 의 결과를 통해 식별할 수 있으며, 이렇게 구성할 경우 영업시간에 대해서도 알 수 있기 때문에 다음과 같이 구성하였다.

(address) & (store\_name) : 각각 주소 , 편의점 이름이다.

<**products**> : product\_upc(PK), vendor\_id(FK),product\_name, brand, package\_type, price, size

(product\_upc) : 물건마다 가지고있는 UPC 를 통해 entity 를 구별할 수 있다. 뿐만 아니라 명세서에 명시한 대로 물건에 대한 다양한 내용들을 속성으로 지정했다.

(vendors\_id) : FK 로 가지고 온다. 이를 통해 물건을 제공한 vendor 조회가능하다.

(product\_name) : 물건의 이름

(brand) : 물건의 브랜드

(package\_type) : 해당 물건이 어떤 type 인지

(price) ( size ) : 물건의 가격 & 사이즈

< Inventory > : store\_id(FK), product\_upc(FK) , current quantity, reorder\_threshold, reorder\_quantity.

(store\_id , product\_upc)(PK) : Weak entity set 으로 구성했는데 , store 과 product 두 entity 에 모두 종속적으로 참여합니다. 따라서 inventory 는 store\_id 와 product\_upc 를 PK 로가집니다.

(reorder\_threshold) : 물건이 해당 점포에 있는 재고 개수와 해당 물건에 대한 threshold 를 설정했습니다.

(reoder\_quantity) : 해당 물건이 임계치에 떨어진다면 reoder\_quantity 를 이용해 자동으로 추가 주문을 할 수 있습니다.

(current\_quantitiy) : 현재 가지고 있는 물건의 양

(recent\_order\_history) : 최근 발주한 날짜

(제가 구성한 ER 다이어그램에서는 저희가 알 수 없는 물건은 팔지 않는 것을 전제로 작성했습니다.)

<vendors> : vendor\_id(PK), vendor\_name, contact\_informantion

(Vendoer\_id) : PK 로써 vendor 를 unique 하게 구분할 수 있다.

(vendor\_name): vendor 의 이름에 대한 정보

(contact\_information) : 아직 vendor 에 추가적인 정보가 contact\_information 라는 속성으로 설정해 두었는데(phone\_number정도로 가정함.) 만약 정보가 많거나 필요한 경우 따로 entity 를 만들어 구성하려고 합니다.

<Sale> : sale\_id(PK), customer\_id(FK) , store\_id(FK), payment\_method, total\_money , datetime

Sale 은 지점에서 customer 가 물건을 구매한 경우를 기록합니다. 물건에 대한 정보와 가격,수량 같은 경우는 sale\_detail entity 를 가지고 따로 구성했습니다. 하지만 ,total\_money 같은 경우는 sale 의 속성으로 남겨두었는데, 이는 매점 점포마다 총 매매출 구할 때 조금 더 용이하기 위해서 남겨두었습니다.

(sale\_id) : PK 로써 이를 통해 어떤 거래를 했는지 UNIQUE 하게 찾을 수 있습니다.

(customer\_id) & (store\_id) : 모두 FK 로써, 고객이 여러 번 구매할 수 있고 , 한 편의점에서 여러 번 구매가 나타날 수 있기 때문에 이렇게 구성하였습니다.

(payment\_method) : 결제 방식

(total\_money) : 한번 할 때 결제량

(datetime): 결제 날짜

<Sale\_detail> : (sale\_id(FK) , product\_upc(Fk))(PK) , quantity

(Sale\_detail) 도 weak entity 로 설정하였습니다. 만약 물건이 팔린 경우 물건이 팔린 id ( sale\_id ) 와 물건 고유 번호(upc) 과 수량(quantity) 를 저장합니다. 이후에 product 가 팔린 수량을 보려고 한다면 , product -> sale\_detail 을 참조해서 검색 할 수 있도록 구성했습니다.

<Customer> : cutomer\_id(PK) , customer\_name , phone , email , membership\_point

Customer 는 고유 id 값을 받아 식별할 수 있습니다. 이때 ,이름,번호,email 같은 정보를 저장할 수 있습니다. 저는 membership\_point 를 속성으로 두었는데 이를 활용해서 비회원/회원 구분을 할 수 있을 뿐만 아니라

( if membership\_point < threshold -> 비회원) vip 같은 경우는 membership\_point 가 높은 경우로 찾아낼 수 있습니다. 만약 수동으로 vip 회원을 지정하고 싶은 경우에는 membership\_point 에 매우 큰 값을 주면 됩니다.

1. Relation

(purchase) : customer 과 sale 같은 경우 1:N 으로 구성하였습니다. 한명의 custormer 가 여러 번 구매할 수 있기 때문에 다음과 같이 구성하였습니다. 또한 sale 이 customer\_id 를 (FK) 로 가지게 됩니다.

(process): Sale 과 store 는 N : 1 로 구성하였습니다. 이는 하나의 매장에서 여러 번 구매할 수 있기 때문입니다. 또한 또한 sale 이 store\_id 를 (FK) 로 가지게 됩니다.

(manges\_inventory) : Inventory 는 weak\_entity 로 설계하였습니다. 따라서 product 와 store 에 모두 전체 참여합니다. 또 둘다 모두 1(inventory):N 의 관계를 가집니다. 하나의 편의점에서 여러 개의 inventory 를 가질 수 있습니다

(invent\_product\_detail) : 또, 하나의 제품은 여러 inventory 에 있을 수 있기 때문에 1:N 관계를 가집니다.

(supply): 여러개의 물건을 vendor 가 제공할 수 있고, 하나의 물건은 하나의 vendor 가 제공한다고 가정하고 작성했으므로 1:N 관계를 가집니다.

(Sell\_product): Sale\_detail 을 weak entity set 으로 설정하였습니다.따라서 products 와 sale 의 (PK) 를 가지고 옵니다. 또, sale 과 product 와 모두 1:N 관계를 가지고 있습니다.

**D. Sample query requirement**

1. **Product Availability**

**Inventory 에서 product\_id(FK) 와 store\_id(FK) 를 확인할 수 있기 때문에 (product 와 store 에 대한 정보를 가지고 있기 때문에) 어떤 매장에 물건이 있는지 , 재고가 얼마나 있는지 확인할 수 있다.**

1. **Top-Selling Items**

**sale table 에서 datatime 을 기준으로 sale\_detail 에서 상품의 개수를 조회해서 구할 수 있으므로 , 지난 한달동안 가장 많이 팔린 제품 검색이 가능하다.**

1. **Store Performance**

**Sale table에서 Datetime 으로 분기(기준 날짜들)를 설정한 뒤에 해당 범위(선택한분기)안에서 store\_id 를 통해 매장에서 total\_money 를 더한뒤 총 결과값을 비교하면 가장 많은 매출을 올린 매장을 알 수 있다.**

1. **Vendor Statistics**

**sale\_detail 에서 각 물건이 팔린 수량을 계산한 뒤 해당 물건에 대해서 product\_upc 를 이용해서 product 까지 도달할 수 있고 그 뒤 product 에 FK 인 vendor\_id 를 통해서 해당 벤더를 찾을 수 있다. 따라서 가장 많은 제품을 판 벤더와, 제품의 판매량까지 알 수 있다.**

**+) 조교님 : reorder 자체를 테이블에서 데이터로 따로 관리할지, 아니면 on-demand로 요청이 들어올때마다 연산을 수행할지를 고민하셔서 실제 상황에 어떤게 더 효율적으로 사용될지 고민하시고 그 이유와 함께 레포트에 적어주세요.**

**이에 대한 답변을 고민하다. 저는 reorder 라는 테이블을 만들면 데이터 베이스 구성이 너무 복잡해 져 정규화에 조금 힘들 것 같다고 판단해 on-demand 로 요청이 들어올때마다 그때그때 처리하는 것으로 선택했습니다.**

1. **Inventory Reorder Alerts**

**Inventory entity 속성안에 있는 current\_quantity 를 통해서 각 매장의 현재 수량을 알 수 있고, 현재수량이 reorder\_theshold 보다 작은 경우를 검색함으로써 해당 제품 수량의 임계값 이하의 제품을 알아낼 수 있다.**

1. **Customer Purchase Patterns**

**Member\_ship point 가 일정 기준이상이면 loyalty\_customer 이라고 설정한 뒤, 해당 customer의 PK 를 가지고 sale에서 같은 datetime 마다 sale\_detail 을 조회한뒤 커피와 같이 물건을 산 경우를 각 물건마다 카운트 해서 따로 저장한다. 조회를 마치면 고객이 “커피” 와 “같이” 산 물건들의 개수를 모두 확인할 수 있고 상위 3개 제품을 선택하면 된다.**

1. **Franchise vs. Corporate Comparison  
   store 의 속성에 ownership\_type 을 저장해 두었는데 이를 기준으로 프랜차이즈 매장을 선택한 뒤, 매장별로 inventory 를 조회해서 파는 물건의 종류를 구하면된다. 이를 직영점과 비교하기 위해서 직영점 역시 마찬가지 방법으로 조회한 뒤 둘을 비교하면 된다.**

텍스트, 도표, 스크린샷, 평행이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.cf. ER schema ( 나중에 필요해질 것 같아 혹시몰라 미리 그려두었습니다.)