CSE4110-01 데이터베이스시스템

[Project 01]
Design ERD for
Conveneince Store System Chain

20200271 안서진

Ⅰ. 문제 개요

편의점 유통 체계 시스템을 처음부터 설계하는 상황을 가정하여 ER 다이어그램을 작성하였다. 처음에는 제품, 매장, 고객, 거래, 공급업체 등 핵심 데이터를 중심으로 엔터티를 나눈 뒤, 이들 간의 관계를 파악하였다. 매장별 재고 현황을 표현하기 위해 Inventory 관계를 설정하였고, 하나의 거래에 여러 제품이 포함될 수 있음을 고려하여 sales_product 관계를 추가하였다. 추가로 membership에 가입한 고객 식별, 프랜차이즈 여부 등 다양한 질의 요구에 대응할 수 있도록 관련 속성과 제약 조건을 보완하였다. 전체적으로는 Query에 대응할 수 있는 가능성, 올바른 ER Diagram 설계 기법 등을 모두 고려하여 다시 설계하는 시간을 가졌다.

Ⅱ. Sample query 를 이용한 설계 과정

편의점 유통 체계를 효과적으로 설계하기 위해, 먼저 과제에서 제시한 7가지 Sample Query를 면밀히 분석하는 것에서 출발하였다. 각 Query가 요구하는 정보를 정확히 도출할 수 있도록, 필요한 entity는 무엇인지, 어떤 attribute와 relationship이 존재해야 하는지를 중심으로 고민하였다.

1. Product Availability

"Which stores currently carry a certain product (by UPC, name, or brand), and how much inventory do they have?"

- 현재 어떤 매장에서 **특정 제품이 있는지** / 그 제품의 **재고**는 얼마나 되는지

해당 질의를 처리하기 위해 필요한 엔터티는 제품(Products), 매장(Stores), 그리고 이 둘을 연결하는 재고 정보(Inventory)이다. Products 엔터티에는 UPC, name, brand 등의 속성이 포함되며, 그중 UPC를 제품을 식별하는 primary key로 설정하였다. 매장은 Stores라는 이름의 엔터티로 설계하였고, store_id를 필수 속성으로 포함하여 primary key로 지정하였다. 두 엔터티를 연결하기 위해 중간 엔터티인 Inventory를 생성하였으며, 여기에는 store_id와 UPC를 foreign key로 설정하였다. 추가적으로 quantity와 minimumStock 속성을 포함하여, 매장별 제품의 재고 수량 및 재주문 기준 정보를 함께 관리할 수 있도록 초기 설계를 구성하였다.

2. Top-Selling Items

"Which products have the highest sales volume in each store over the past month?"

- 지난 한 달 동안 각 매장에서 가장 많이 판매된 상품

2번 질의를 처리하기 위해 필요한 엔터티는 판매 기록(salesTransaction)과 개별 거래 항목(sales_product), 그리고 제품(Products)이다..
판매기록(salesTransaction)에는 각 거래의 고유 식별자(id), 거래일(salesDate), 매장 정보(store_id) 등의 속성을 포함시켰고, id는 primary key로 설정하였다. 거래에 포함된 제품과 수량 정보는 sales_product 엔터티에 저장했고, transaction_id와 UPC를 foreign key로 포함하였다. 제품 식별을 위해 UPC는 Products를 고유하게 식별할 수 있으므로 단독으로 primary key로 설정하였고, 판매 수량을 나타내는 quantity 속성도 함께 포함하였다. 이를 통해 각 매장에서 특정 기간 동안 어떤 제품이 가장 많이 판매되었는지를 분석할 수 있도록 설계하였다.

3. Store Performance

"Which store has generated the highest overall revenue this quarter?"
- 이번 분기 가장 **매출이 높은** 지점
해당 질의를 처리하기 위해 필요한 엔터티는 판매 기록(salesTransaction), 판매 항목(sales_product), 그리고 제품(Products)이다. 1번과 2번 질의를 통해 세 개의 엔터티를 만들어두었으므로, 엔터티는 추가하지 않았다. Products에 포함된 price 속성과 sales_product의 quantity, salesTransaction의 salesDate, store_id 속성을 조합함으로써 매장별 총 매출을 계산할 수 있도록 설계하였다. Products에는 price 속성이 존재하고 sales_product에는 quantity 속성이 존재하는데, 이 두 값을 곱하면 각 거래의 매출을 추출할 수 있다.

4. Vendor Statistics

"Which vendor supplies the most products across the chain, and how many total units have been sold?"
- 편의점 체인 전체에서 가장 많은 제품을 공급한 **공급업체**는 누구이며, 그 제품들은 **총 몇 개**가 판매되었는지

편의점 체인 전체에서 가장 많은 제품을 공급한 공급업체와, 그 제품들이 판매된 갯수를 처리하기 위해서는 공급업체(Vendor), 제품(Products), 그리고 판매 항목(sales_product) 엔터티가 필요하다. Vendor 엔터티에는 고유 식별자(id, PK) , 업체명(name), 연락처(phone) 등을 포함하였다. 제품 엔터티인 Products에는 vendor_id를 foreign key로 받아와서 각 제품이 어느 공급업체로부터 제공되었는지 알 수 있도록 하였다. sales_product의 quantity 속성을 통해 판매 수량을 확인할 수 있다. Vendor, 즉 공급업체 단위로 제품이 판매된 갯수를 확인하기 위해서는 Products 엔터티와 연결하면 된다.

5. Inventory Reorder Alerts

"Which products in each store are below the reorder threshold and need restocking?"
- 매장에서 **재고**가 재주문 기준 이하로 떨어져서 보충이 필요한 상품

5번 쿼리에 대해 답변하기 위해서는 매장(Stores), 제품(Products), 그리고 이 둘을

연결하는 Inventory 엔터티가 요구된다. '재주문 기준 이하'인지 여부를 파악하기 위해서 '재주문 기준'을 db에 저장해야하며, 따라서 Inventory 엔터티에 minimumStock 속성을 추가하였다. 이로써 각 제품의 재고 수량(quantity)이 기준치(minimumStock)보다 낮은 경우를 처리할 수 있다. Products는 또한 Stores의 ID값과 Products의 UPC 값을 foreign key로 받아와야하고, 두 값이 복합 PK로 설정하였다. 해당 엔터티와 속성등을 통해 매장별로 재고가 부족한 제품 목록을 파악할 수 있다.

6. Customer Purchase Patterns

"List the top 3 items that loyalty program customers typically purchase with coffee."

- **멤버십**(로열티 프로그램) 고객들이 커피와 함께 보통 구매하는 상위 3가지 **상품**

6번 쿼리를 해결하기 위해서는 고객(Customer), 거래 내역(salesTransaction), 판매 항목(sales_product), 그리고 제품(Products) 엔터티가 필요하다. Customer 엔터티에는 고객을 식별하는 customer_id와 membership 여부를 나타내는 속성이 포함되며, customer_id를 통해 Customer을 고유하게 식별할 수 있다. 거래내역을 저장하는 salesTransaction 엔터티에는 customer의 ID를 foreign key로 포함시켜 고객과 거래 간 연결을 가능하게 하였다. sales_product을 통해 물건 판매 항목 정보를 저장하였고, 각 제품은 Products의 UPC를 참조한다.

7. Franchise vs. Corporate Comparison

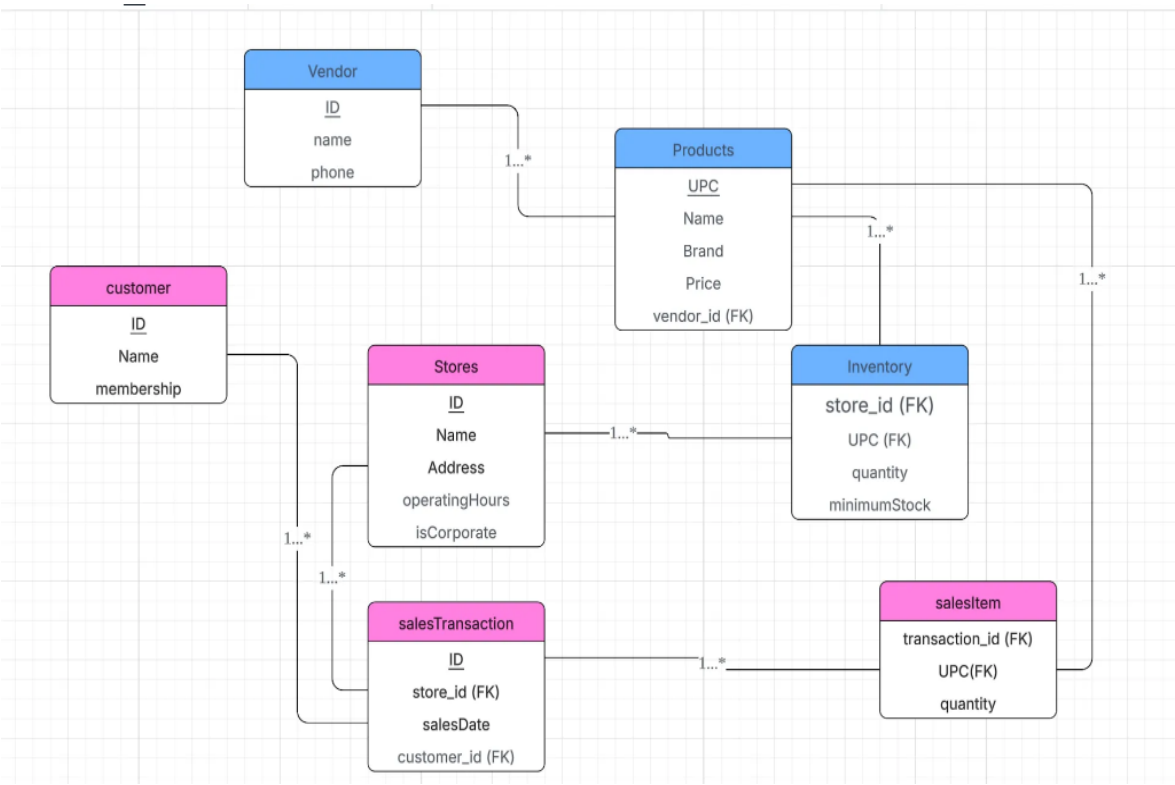
"Among franchise-owned stores, which one offers the widest variety of products, and how does that compare to corporate-owned stores?"

- **프랜차이즈 매장** 중 가장 다양한 **상품**을 제공하는 **매장**은 어디이며, 이는 **본사 직영 매장**과 어떻게 비교되는지

해당 질의를 처리하기 위해 Stores, Products, 그리고 두 엔터티 간의 M:N 관계를 나타내는 중간 엔터티인 Inventory 엔터티가 요구된다. Stores 엔터티에는 매장이 프랜차이즈인지 본사 직영점인지, 즉 매장이 운영되는 형태를 구분하여 저장할 수 있도록 isCorporate 속성을 저장하였다. Inventory는 각 매장이 어떤 제품을 얼마나

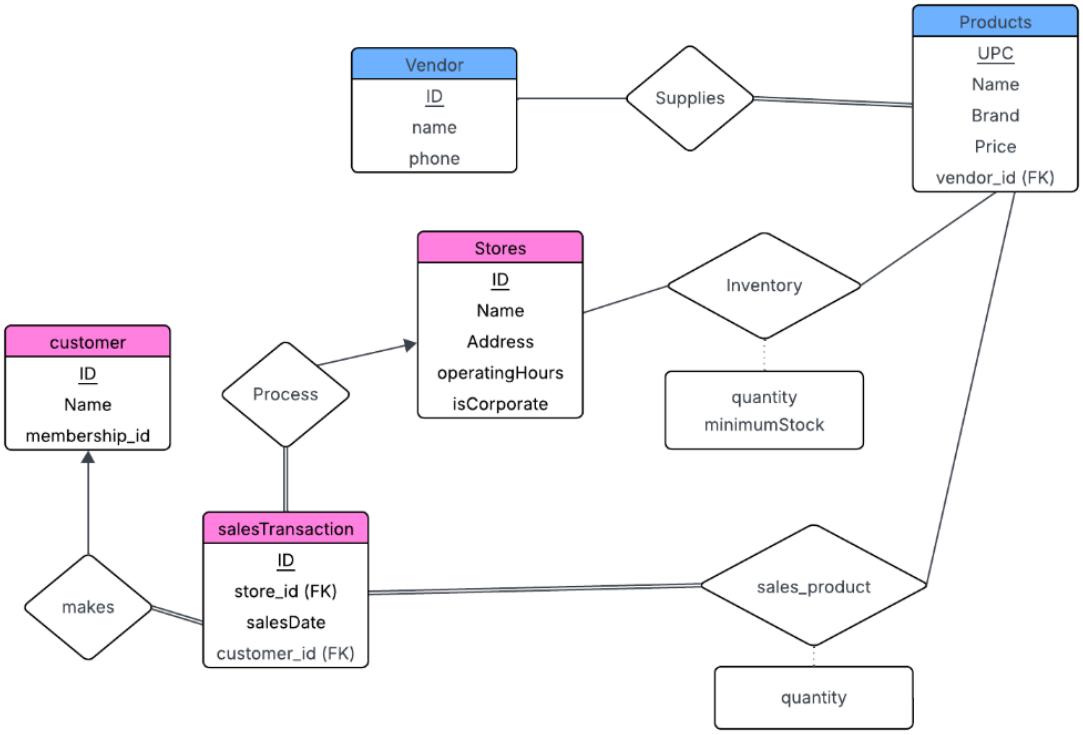
보유하고 있는지를 나타내고 있는데, Stores와 Product 엔터티의 store_id와 UPC를 foreign key로 참조하였다. Inventory 엔터티를 기준으로 매장별로 보유하고 있는 제품 수를 계산하였고, isCorporate 속성은 boolean값으로 저장한다.

위의 7가지 질의에 대해 답변을 하는 과정을 거치며 Stores, Products, Inventory, salesTransaction, sales_product, Vendor, Customer 등 필요한 엔터티와 속성을 추가하였다. 추가적인 엔터티나 속성 없이 엔터티들간의 결합을 통해 쿼리를 해결할 수 있는 경우가 있는지도 확인하는 과정을 거쳤고, 초안 설계안이 다음과 같이 그려졌다. 이를 기반으로 최종 ERD를 구상하였다.



Ⅲ. 최종 ER Diagram

- About Entity, Relationship, Mapping Cardinality



1. Entity 설계 근거

ⅰ) Customer

- 편의점 이용 고객 정보를 포함하는 엔터티

고객 식별을 위한 ID를 primary key로 설정하였고, 이름(Name)과 멤버십 여부를 나타내는 membership 속성(membership_id) 을 포함하였다. Customer 엔터티는 고객별 구매 내역 분석을 위해 설계되었으며, 특히 이번 보고서에서는 6번(Customer Purchase Pattern) 질의 해결에 사용된다. 또한 salesTransaction(판매 내역) 과의 관계를 통해 특정 고객의 거래 내역을 조회할 수 있다.

ⅱ) Stores

- 편의점 매장 정보를 포함하는 엔터티

각 매장의 고유 식별자인 ID를 primary key로 설정하였고, 매장명(Name), 주소(Address), 운영 시간(operatingHours)을 속성으로 포함하였다. 편의점 매장의 프랜차이즈와 직영 매장 여부를 구분하기 위해 boolean값을 갖는 isCorporate

속성을 추가하였다. 이 엔터티는 질의 1번(Product Availability), 2번(Top-Selling Items), 3번(Store Performance), 5번(Inventory Reorder Alerts), 7번(Franchise vs. Corporate Comparison) 등 대부분의 질의에서 사용되었다.

ⅲ) Products

- 편의점에서 판매되는 상품 정보를 포함하는 엔터티.

UPC(제품 고유 식별자)를 primary key로 설정하였고, 제품명(Name), 브랜드(Brand), 가격(Price) 등의 추가 속성을 포함한다. 제품을 공급하는 업체(Vendor) 의 ID 값을 참조하는 vendor_id 를 foreign key로 설정하였다. Products 엔터티는 이번 db 설계에서 핵심적인 역할을 한다. 질의 1번(Product Availability), 2번(Top-Selling Items), 4번(Vendor Statistics), 5번(Inventory Reorder Alerts), 6번 (Customer Purchase Patterns), 7번(Franchise vs. Corporate Comparison) 등 다양한 질의의 해결에 사용되었다.

ⅳ) salesTransaction

- 고객의 거래 기록을 저장하는 엔터티 (판매 내역).

salesTransaction의 ID를 설정해 primary key로 이용하였고, 이를 이용해 거래를 구분한다. 추가적으로 거래 일자(salesDate)를 저장하고, 거래 매장(store_id), 고객(customer_id)을 foreign key로 포함하였다. salesTransaction 엔터티를 통해 고객이 어떤 매장에서 어떤 시점에 거래했는지를 기록한다. 본 과제에서는 질의 2번(Top-Selling Items), 3번(Store Performance), 6번 (Customer Purchase Patterns) 을 해결하는데 활용하였고, sales_product와 연결하여개 별 제품 판매 내역 역시 확인할 수 있다.

ⅴ) Vendor

- 제품 공급 업체의 정보를 담는 엔터티.

제품 공급 업체의 식별자인 ID를 primary key로 설정하였으며, 업체명(name)과 연락처(phone) 정보를 포함하였다. 질의 4번(Vendor Statics)을 해결하는데 활용된다.

2. Relationship 설계 근거

ⅰ) makes (Customer - salesTransaction)

Customer와 salesTransaction 엔터티 사이를 연결하는 makes 관계은 고객이 하나 이상의 거래를 수행한다는 점을 나타낸다. 고객은 편의점에서 1회 이상 구매할 수 있기 때문에 Customer와 salesTransaction 사이는 1:M 관계로 나타내었다. 하지만, 하나의 거래는 반드시 하나의 고객에 의해 이루어지므로, salesTransaction은 반드시 하나의 Customer에 종속되기 때문에, salesTransaction은 total participation, Customer는 partial participation으로 설정하였다. 해당 관계는 2번(Top-Selling Items) 질의를 해결하는 데 활용하였다.

ⅱ) Process (Stores - salesTransaction)

Stores와 salesTransaction 엔터티 사이를 연결하는 Process 관계는 특정 거래가 어떤 매장에서 발생했는지를 명시한다. 하나의 매장에서 여러 거래가 발생할 수 있으므로 Stores와 salesTransaction은 1:M 관계로 나타내었다. 모든 거래는 반드시 매장 내에서 발생하기 때문에 salesTransaction은 Stores에 대해 total participation 관계를 가진다고 나타내었고, 반면, 어떤 거래도 발생하지 않은 매장 (ex. 신규 매장)이 존재할 수 있기 때문에 stores쪽에는 partial participation으로 나타내었다. 2번(Top-Selling Items), 3번(Store Performance) 쿼리를 해결할 때 사용하였다.

ⅲ) Inventory (Stores - Products)

Stores와 Products 엔터티를 연결하는 Inventory 관계는 특정 매장이 어떤 제품을 얼마만큼 보유하고 있는지를 나타낸다. 하나의 매장은 반드시 하나의 제품을 취급하는 것이 아니며, 또 하나의 제품이 한 매장에서만 판매되는 것도 아니기 때문에 해당 관계는 M:N (many-to-many) 로 표현하였다. Stores와 Products, 두 엔터티 사이를 효율적으로 연결하고 관리하기 위해 Inventory를 별도의 관계로 두었고, store_id와 UPC를 foreign key로 참조하였다.

해당 Inventory 관계에서는 재고 수량(quantity)과 재주문 기준(minimumStock) 속성을 관계 속성으로 포함하였다. 이를 통해 재고 현황과 각 매장에서 재고가 재주문 기준 이하려 떨어져서 보충이 필요한 상품이 무엇인지에 대해 파악할 수 있는 정보를 저장하였다. 모든 매장이 제품을 반드시 보유하고 있어야 하는 것은 아니며, 상품 중에서도 재고가 없는 상품이 존재할 수 있기 때문에 양쪽 모두 partial participation 이라고 판단하였다. 1번(Product Availability), 5번(Inventory Reorder

Alerts) 질의를 해결하는 데 활용한다.

ⅳ) sales_product (salesTransaction - Products)

salesTransaction과 Products 엔터티를 연결하는 sales_product 관계는 하나의 거래에서 어떤 제품이 얼마나 판매되었는지를 기록하는데 사용된다. 동일한 제품이 여러 거래에서 판매될 수도 있고, 하나의 거래 역시 여러 개의 제품을 포함할 수 있기 때문에 sales_product 관계는 M:N 관계이다. 각 거래 항목에 포함된 제품 수량을 나타내기 위해 quantity를 관계 속성으로 포함하였다. 또한, 모든 거래가 제품을 포함하며, 판매된 제품이 없는 거래는 없다고 가정하였기 때문에, 두 측 모두 total participation으로 설계하였다. 2번(Top-Selling Items), 3번(Store Performance), 4번 (Vendor Statistics) 질의 해결에 활용된다.

ⅴ) Supplies (Vendor - Products)

Vendor와 Products 엔터티를 연결하는 Supplies 관계는 공급업체가 제품을 납품한다는 점을 보여준다. 하나의 공급업체가 여러개의 제품을 공급할 수 있기 때문에 Vendor와 Products는 1:M 관계로 설정하였다. 모든 제품은 공급업체 없이 납품될 수 없기 때문에 Products 측은 total participation이지만, 반면 공급업체는 특정 시점에 제품을 공급하지 않을 수 있기 때문에 Vendor 측은 partial participation으로 나타내었다. 이 관계는 4번 (Vendor Statistics) 질의를 해결하는 데 활용된다.