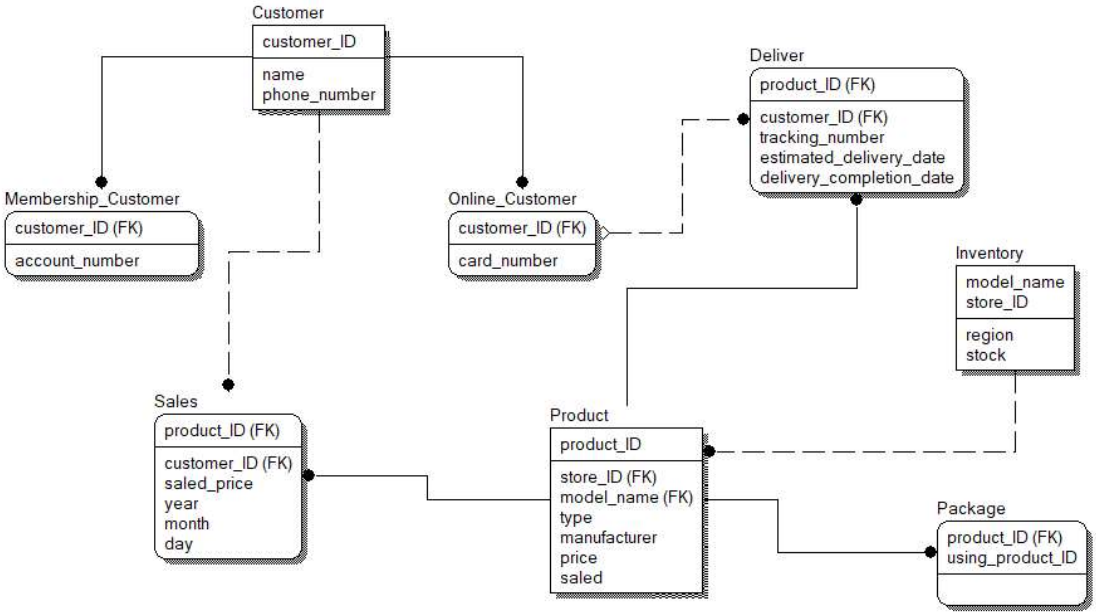
20190554 송재현 DataBase System Project 2

Physical Schema diagram



Customer

customer_ID가 primary key이고 customer_ID->name, phone_number의 functional dependency가 있다. 다른 DB에서는 phone_number를 여러 개 있다고 가정하는 경우도 있으나 보통의 경우 한 사람당 phone_number는 하나라고 가정하기에 여기서는 phone_number는 하나로 가정하였다. 또한 이미 등록된 번호의 경우 다른 사람이 이 번호를 사용할 수 없는 정책 또한 널리 사용되므로 customer_ID -> name, phone_number라는 functional dependency에서 customer_ID가 super key이므로 BCNF를 만족한다.

Membership_Customer

Customer의 Specialization으로 customer_ID가 primary key이고 customer_ID -> account_number의 functional dependency가 존재한다. 따라서 customer_ID가 super key이기 때문에 BCNF를 만족한다.

Online_Customer

Customer의 Specialization으로 customer_ID가 primary key이고 customer_ID -> card_number의 functional dependency가 존재한다. 따라서 customer_ID가 super key이기 때문에 BCNF를 만족한다.

account_number와 card_number의 경우 위와 같이 specialization하지 않으면 온라인 고객이 아니거나 멤버십 고객이 아닌 고객의 경우 NULL값으로 남기 때문에 specialization하도록 구현하였다. name은 같은 경우가 흔히 있으므로 각 customer별로 ID를 부여해서 구분하도록 하였다.

Product
product_ID가 primary key이고 product_ID -> model_name, store_ID, type, manufacturer, price, saled의 fucntional dependency가 있다. 따라서 product_ID가 super key이기 때문에 BCNF를 만족한다.
같은 모델일지라도 그에 해당하는 product_ID, 즉 우리가 아는 serial number는 다르다. 모델명으로만 상품을 구분하기에는 판매되는 상품의 serial number는 저장되지 않으므로 이를 위해 product_ID로 상품을 구분한다. 이미 판매된 상품일지라도 delete의 경우 지원하지 않는 경우도 있기 때문에 saled라는 attribute를 통해 판매된 상품인지 판매되지 않은 상품인지를 구분한다.

Package
package의 product_ID와 using_product_ID가 primary key이다. BCNF를 만족한다.
Product에서 package의 경우 이미 존재하는 상품들의 묶음이므로 이미 존재하는 상품들 중 어떤 것이 쓰였는지를 알기 위해 위의 relation을 만들었다.

Inventory
model_name, store_ID가 primary key이고 model_name, store_ID -> region, stock의 functional depency가 있다. 따라서 model_name, store_ID가 super key이기 때문에 BCNF를 만족한다.
상품은 product_ID로 구분하는 것이 맞으나 전체적인 재고는 모델명으로 구분한다. 모델명과 store_ID로 지점에 따른 상품의 재고를 따로 관리한다.

Deliver
Product_ID가 primary key이고 product_ID -> tracking_number, estimated delivery date, delivery completion date, customer_ID의 functional dependency가 존재한다. 이 때 tracking number -> product_ID, estimated delivery date, delivery completion date, customer_ID의 functional dependency 역시 존재한다. tracking number와 product_ID 모두 super key이므로 BCNF를 만족한다. 이 때 delivery_completion_date는 아직 배송이 완료되지 않았다면 NULL값으로 존재할 수 있다.
product_ID, 즉 상품의 고유번호는 그 상품만 갖게 되므로 이를 통해 배송 정보를 알 수도 있고 송장번호, tracking_number를 통해서도 배송 정보를 전부 알 수 있다. 그 중 product_ID를 primary key로 지정하였다.

Sales
product_ID가 primary key이고 product_ID -> customer_ID, saled_price, year, month, day의 functional dependency가 있다. 이외에는 functional dependency가 없으므로 product_ID가 super key이기 때문에 BCNF를 만족한다. year, month, day를 통해 시간을 확인하고 현재 가격과 판매 시점 당시 판매 가격은 다를 수 있으므로 saled_price를 따로 기록한다. store_ID의 경우 product와의 join을 통해서 파악할 수 있다.

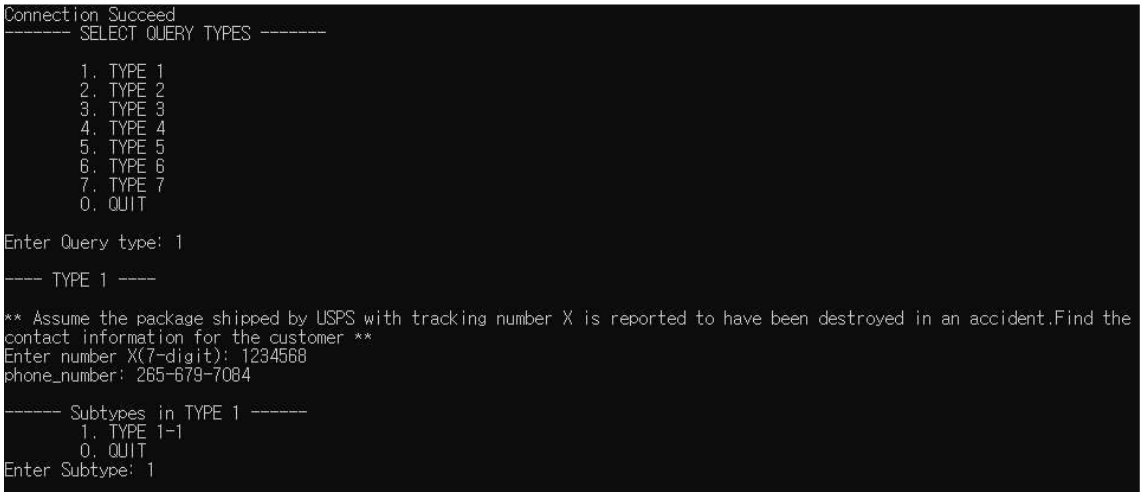
Project1에서 만든 relation들에서 attribute들이 바뀌고 내용을 바꾼 relation은 있으나 원

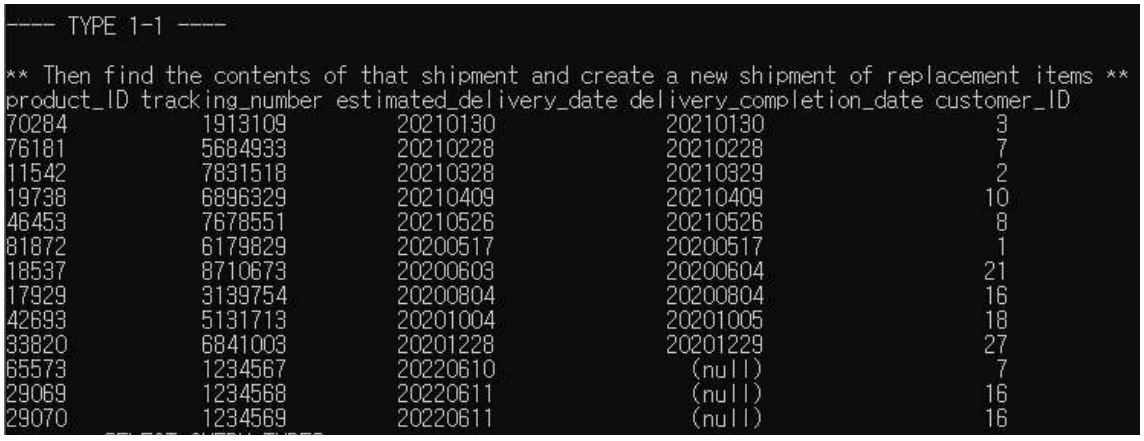
래 있던 relation들이 모두 BCNF여서 따로 BCNF로 decompose할 필요가 없었다.

MySQL ODBC implementation

실행 결과

Query 1





Query 2

```
Enter Query type: 2

---- TYPE 2 ----

**  Find the customer who has bought the most (by price) in the past year **
customer_ID and total_bill: 1 8029.00

------ Subtypes in TYPE 2 ------
        1. TYPE 2-1
        0. QUIT
Enter Subtype: 1

---- TYPE 2-1 ----

** Then find the product that the customer bought the most **
customer_ID model_name unit_sale
1               A7c       2
```

Query 3

```
---- TYPE 3 ----

** Find all products sold in the past year **
product_ID, model_name
70284       24MP58VQ
71328       24MP58VQ
11111       24MP58VQ
76181       S24R35A
11542       MWPF2KHA
19738       NX300
46453       EOS200D
60895       A7c
60896       A7c
37779       A2482
65363       F100N
22222       F100N
22278       L6290
81662       M706n
70284       27MP58VQ
60896       A10c
```

```
---- TYPE 3-1 ----

** Then find the top k products by dollar-amount sold **
which K?: 5
model_name, total_sold
A7c          5900.00
A10c         2950.00
A2482        1090.00
F100N        960.00
EOS200D      700.00
```

```
---- TYPE 3-2 ----

** And then find the top 10% products by dollar-amount sold **
model_name, total_sold
A7c          5900.00
------- SELECT QUERY TYPES -------

        1. TYPE 1
        2. TYPE 2
        3. TYPE 3
        4. TYPE 4
        5. TYPE 5
        6. TYPE 6
        7. TYPE 7
        0. QUIT
```

Query 4

```
---- TYPE 4 ----

** Find all products by unit sales in the past year **
product_ID, model_name, unit_sale
 70284      24MP58VQ         3
 60895           A7c         2
 65363         F100N         2
 76181        S24R35A         1
 11542       MWPF2KHA         1
 19738          NX300         1
 46453        EOS200D         1
 37779          A2482         1
 22278          L6290         1
 81662          M706n         1
 70284      27MP58VQ         1
 60896           A10c         1
```

```
---- TYPE 4-1 ----

**  Then find the top k products by unit sales **
which K?: 5
product_ID, model_name, unit_sale
 70284      24MP58VQ         3
 65363         F100N         2
 60895           A7c         2
 76181        S24R35A         1
 11542       MWPF2KHA         1
```

```
---- TYPE 4-2 ----

** And then find the top 1000007FF71797BC00roducts by unit sales **
product_ID, model_name, unit_sale
 70284      24MP58VQ         3
```

Query 5

```
** Find those products that are out-of-stock at every store in California **
model_name
27MP58VQ
S23R35A
MWPF10KHA
```

Query 6

```
---- TYPE 6 ----

** Find those packages that were not delivered within the promised time **
product_ID
18537
42693
33820
```

Query 7

```
---- TYPE 7 ----

** Generate the bill for each customer for the past month **
customer_ID, monthly_bill
1            165.00
11           290.00
12           190.00
15           710.00
8            2960.00
```

Query 설명

Query 1의 경우 tracking number가 X인 배송이 중간에 사고로 유실된 경우 customer의 연락처를 구하는 query이다. X를 stdin으로 입력받아 tracking_number로 저장하고 이를 query문에 넣는다. deliver 테이블에서 tracking_number가 X인 customer_ID를 구하고 customer 테이블에서 이 customer_ID를 찾아 phone_number를 구한다.

이후 1-1은 원래 배송되던 상품의 정보를 파악하여 새로운 대체품을 배송시켜 이에 대한 정보를 만드는 것이었다. 여기서 tracking_number로 원래 있던 product_ID를 query문을 이용해 구한 뒤 기존의 product_ID와 다른 새 상품을 product relation에 추가시켰다. 동일한 모델의 product_ID가 다른 새 제품을 tracking_number를 통해 deliver의 정보들을 구한 뒤 이를 이용해 다시 새로운 shipment를 만들 수 있었다. 이 때 new shipment에 들어가는 product_ID와 tracking_number를 기존의 그것에 1을 더하는 방식을 통해서 만들었는데 이미 product_ID나 tracking_number가 있을 경우 다른 ID와 number와 겹치지 않을 때까지 더해서 만들어줬다. estimated_delivery_date는 기존과 똑같이 작성하였고 delivery_completion_date는 NULL로 할당해주었다. customer_ID는 tracking_number를

통해 구할 수 있었다.

Query2의 경우 작년에 가장 돈을 많이 사용한 고객을 찾는 것이 문제였는데 customer_ID로 group을 형성하고 saled_price의 sum으로 내림차순 정렬하여 가장 위에 있는 tuple을 구함으로 이를 해결할 수 있었다. Query2-1의 경우 이 고객이 가장 많이 산 물건을 찾는 것이었는데 2에서 구한 table에서 customer_ID를 구한 후 count(model_name)으로 정렬하여 customer_ID가 위에서 구한 ID이며 내림차순으로 정렬한 tuple 중 가장 위의 tuple에 해당하는 모델명이 가장 많이 구매한 모델임을 확인할 수 있었다.

Query3의 경우 작년에 판매된 모든 물건을 찾는 것이었는데 sales와 product의 join에서 year가 2021년이고 sales와 product의 product_ID가 같은 tuple을 골라서 원하는 테이블을 얻을 수 있었다. 3-1의 경우 금액적으로 가장 많이 판매된 상품 k개를 구하는 것이었는데 k는 입력 받은 수이다. 위에서 구한 테이블과 비슷하게 year가 2021년이고 sales와 product의 product_ID가 같은 tuple을 model_name으로 group 지어서 saled_price의 sum을 구하고 이를 기준으로 정렬하였다. 이 때 내림차순으로 정렬하고 위의 있는 k개의 tuple을 통해서 원하는 정보를 얻을 수 있었다. 3-2의 경우 위의 테이블에서 model명의 개수를 세서 이 중 상위 10프로에 해당하는, 즉 개수에 1/10을 곱하게 되면 3-1과 별반 다를바 없이 금액적으로 가장 많이 판매된 상품을 구할 수 있었다.

Query4의 경우 작년에 판매된 모든 물건의 unit sales를 파악하는 것이었는데 model_name으로 grouping하여 count(model_name)을 의미하는 unit_sale을 구하고 이를 이용해 정렬하였다. 4-1의 경우 위의 테이블에서 상위 k개를 구하면 됐고 4-2의 경우에도 위의 테이블에서 모델명의 개수 중 10%에 해당하는 tuple을 계산하여 구할 수 있었다.

Query5의 경우 Inventory에서 region이 California이고 stock이 0인 제품들을 모두 구하였다.

Query6의 경우 product에서 type이 package이고 deliver와의 join에서 delivery_completion_time에서 estiamted_delivery_time을 빼서 이 값이 0보다 크면 예정 배송일보다 늦게 배송됐다는 의미이므로 이 조건들을 통해 구할 수 있었다.

Query7의 경우 sales에서 2022년 5월에 해당하는 tuple 중에서 Customer_ID로 grouping하여 saled_price의 합을 구하였고 구매 기록이 있는 사람들에 한해서 sum(saled_price)를 계산하여 each customer에게 bill을 제공할 수 있었다.