DATABASE SYSTEM

Project2. BCNF Decomposition, Physical Schema Diagram and Queries

Electronics Vendor company

CSE4110-01

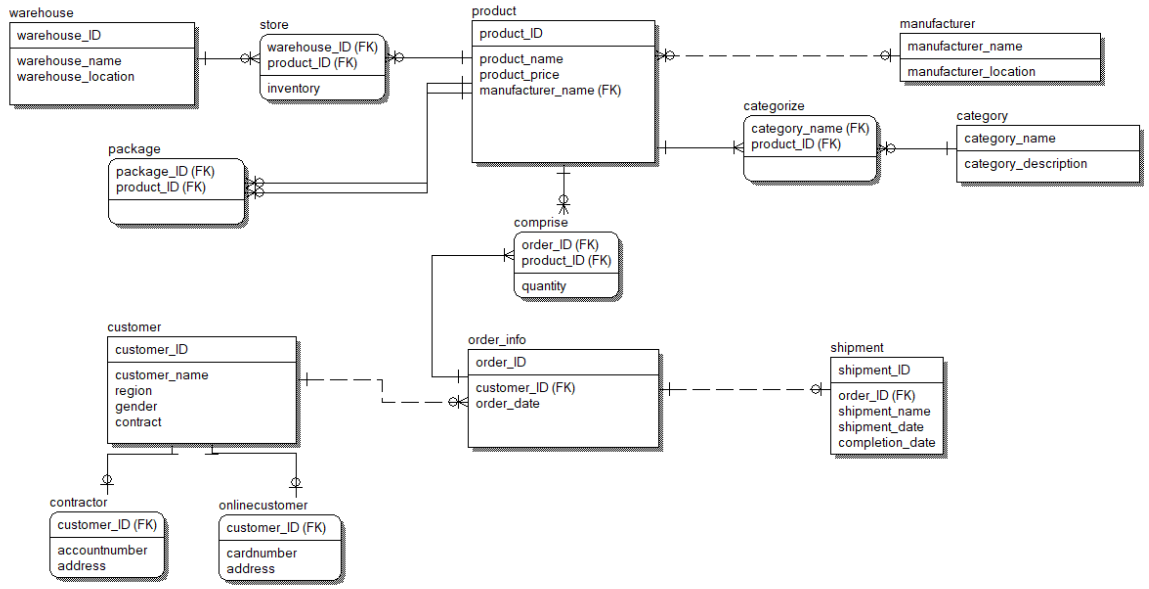
20160530 박상엽

1. 프로젝트 개요

가상의 Electronics Vendor company의 DBA라고 가정한다. 이 회사에서는 회사 운영의 기초가 되는 데이터베이스의 주요 부분을 재설하기로 결정했고, 설계 제안을 요청하는 관리자는 컴퓨터에 대해 잘 알지 못하고, 기술 수준에서 매우 상세한 사양을 제공할 수 없다. 때문에 요구사항을 분석하고 주어진 질의에 적절하게 대응할 수 있도록 데이터베이스 설계 제안서를 작성해야 한다.

2. BCNF Decomposition

2.1 Decomposed Logical Schema Diagram



2.2 Description

1) Product {product_ID -> product_name, product_price, manufacturer_name}

   상품의 ID가 같으면 상품명, 상품가격, 제조사에 대한 값도 동일하므로 종속성이 존재한다. Product_ID가 PK이므로 super key이고 조건을 만족하여 BCNF를 만족한다.

2) Warehouse {warehouse_ID -> warehouse_name, warehouse_loaction}

   창고의 ID가 같으면, 창고의 이름과 위치가 동일하여 함수적 종속성이 있다. Warehouse_ID가 PK이므로 super key이고 조건을 만족하여 BCNF를 만족한다.

3) Store {warehouse_ID, product_ID -> inventory}

   창고의 ID와 상품의 ID가 동일하면, 상품의 개수도 동일하다. 따라서 함수적 종속성이 존재하며 warehouse_ID와 product_ID가 PK이므로 super key이고 조건을 만족하여 BCNF를 만족한다.

4) Manufacturer {manufacturer_name -> manufacturer_location}

제조사가 같으면 제조사의 위치가 같으므로 함수적 종속성이 존재한다. 따라서manufacturer_name이 PK이므로 super key이고 조건을 만족하여 BCNF를 만족한다.

5) Category { category_name -> category_description}

카테고리명이 같으면 그에 따른 설명도 동일하므로 함수적 종속성이 존재한다. Category_name이 PK이므로 super key이고 조건을 만족하여 BCNF를 만족한다.

6) order_info{ order_ID -> customer_ID, order_date}

주문 ID가 같으면, 고객의 ID와 주문일이 동일하다. 따라서 함수적 종속성이 존재한다. Order_ID가 PK이므로 super key이고 조건을 만족하여 BCNF를 만족한다.

7) customer{customer_ID -> contract, name, gender, region}

고객의 ID가 같으면, 연락처, 이름, 성별, 지역이 동일하다. 따라서 함수적 종속성이 존재한다. Customer_ID가 PK이므로 super key이고 조건을 만족하여 BCNF를 만족한다.

8) contractor {customer_ID -> account, address}

specialization을 통해 customer_ID를 받아왔다. 고객의 ID가 같으면, 계좌정보와 주소가 동일하다. 따라서 함수적 종속성이 존재하고, customer_ID가 PK이므로 super key이고 조건을 만족하여 BCNF를 만족한다.

9) onlinecustomer {customer_ID -> cardnumber ,address}

specialization을 통해 customer_ID를 받아왔다. 고객의 ID가 같으면, 계좌정보와 주소가 동일하다. 따라서 함수적 종속성이 존재하고, customer_ID가 PK이므로 super key이고 조건을 만족하여 BCNF를 만족한다.

10) Categorize

다대다 관계에 의해 생성된 테이블로 category_name과 product_name이 PK이므로 super key이다. 따라서 BCNF를 만족한다.

11) package

다대다 관계에 의해 생성된 테이블로 package_ID와 product_ID가 PK이므로 super key이다. 따라서 BCNF를 만족한다.

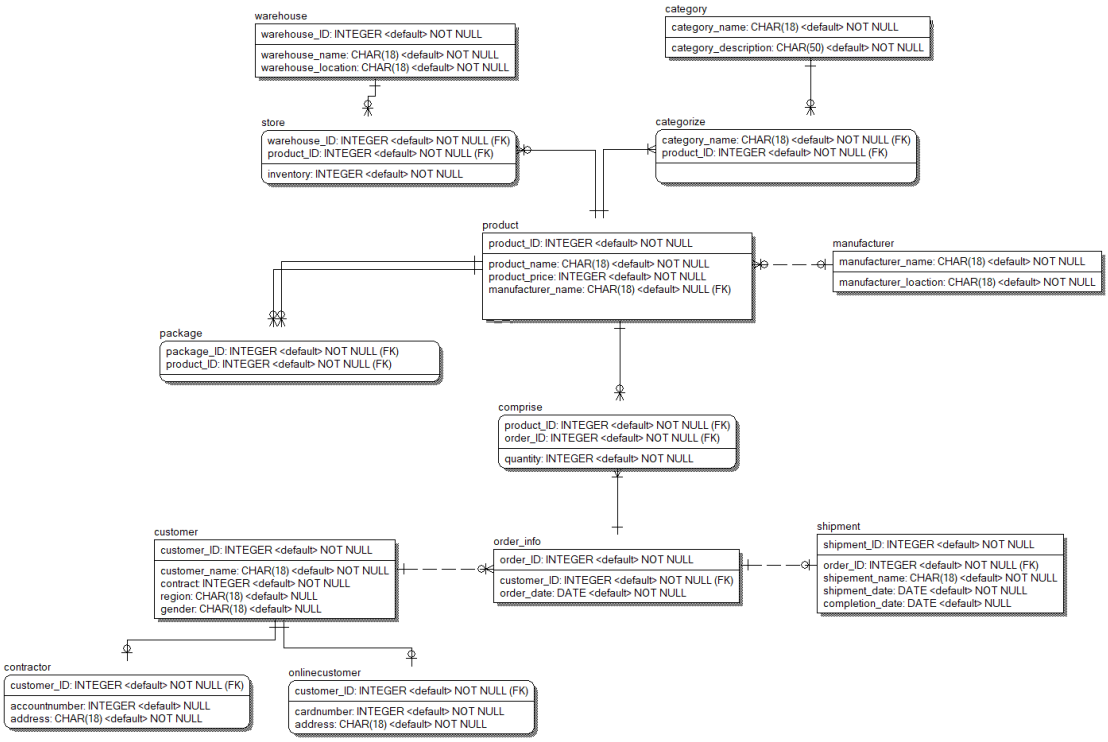
12) comprise {order_ID, product_ID -> quantity}

다대다 관계에 의해 생성된 테이블로 order_ID, product_ID가 PK 역할을 하며, 둘의 값이 동일하면 수량 또한 동일하여 함수적 종속성이 존재한다. 따라서 BCNF를 만족한다.

13) Shipment {shipment_ID -> order_ID, shipment_name, order_date, completion_date}

3. Physical Scheme diagram

3.1 Decomposed Logical Schema Diagram



3.2 Description

Entity	Attribute	Description
product	product_ID	정수형이며 PK이므로 not null이다.
	product_name	문자형이며 not null이다.
	manufacturer_name	Manufacturer을 참조하여 FK이며 따라서 문자형이다. 패키지 상품의 경우 제조사가 구분되지 않기 때문에 null을 허용한다.
package	package_ID	product 참조, 다대다 관계에 의한 형성된 PK이자 FK이다. 따라서 product_ID의 정수형을 가져오며 not null이다.
	product_ID	
manufacturer	manufacturer_name	문자형이며 PK이므로 not null이다.
	manufacturer_location	문자형이며 not null이다.
warehouse	warehouse_ID	정수형이며 PK이므로 not null이다.
	warehouse_name	문자형이며 not null이다.
	warehouse_location	문자형이며 not null이다.
store	warehouse_ID	Warehouse를 참조하며 PK이자 FK이다. 따라서 정수형이며 not null이다.
	product_ID	product 참조하며 PK이자 FK이다. 따라서 정수형이며 not null이다.
	Inventory	상품의 수량을 측정한다. 정수형이며 not null이다.
category	category_name	문자형이며 PK이므로 not null이다.

	category_description	문자형이며 not null이다.
categorize	category_name	category PK이자 FK이다. 따라서 문자형이며 not null이다.
	product_ID	product 참조하며 PK이자 FK이다. 따라서 정수형이며 not null이다.
order_info	order_ID	정수형이며 PK이므로 not null이다.
	customer_ID	Customer을 참조하며 FK이다. 따라서 정수형이며 not null이다.
	order_date	date형이며 not null이다.
shipment	shipment_ID	정수형이며 PK이므로 not null이다.
	order_ID	Order_info를 참조하며 FK이다. 따라서 정수형이며 not null이다.
	shipment_name	정수형이고 not null이다.
	shipment_date	date형이며 not null이다.
	completion_date	date형이며 배송 완료가 된 이후에 업데이트하므로 null을 허용한다.
customer	customer_ID	정수형이며 PK이므로 not null이다.
	contact	정수형이며 not null이다.
	customer_name	고객의 이름으로 문자형이며 not null이다.
	region	문자형이며 필수 정보는 아니므로 null을 허용한다.
	gender	문자형이며 필수 정보는 아니므로 null을 허용한다.
contractor	customer_ID	Customer을 참조하며 PK이자 FK이다. 따라서 정수형이며 not null이다.
	accountNumber	정수형이며 not null이다.
	address	문자형이며 not null이다.
online customer	customer_ID	Customer을 참조하며 PK이자 FK이다. 따라서 정수형이며 not null이다.
	cardNumber	정수형이며 not null이다.
	address	문자형이며 not null이다.

4. ODBC implementation

4.1 CRUD

1) Create and Insert

```
FILE* fp = fopen("20160530.txt", "r");

fseek(fp, 0, SEEK_END);
size = ftell(fp);
buffer = (char*)malloc(size + 1);
memset(buffer, 0, size + 1);
fseek(fp, 0, SEEK_SET);
fread(buffer, size, 1, fp);
```

```
}
int state = 0;
printf("₩n** Create table and insert tuple **₩n");
const char* create = strtok(buffer, ";");
while (create != NULL) {
    state = 0;
    state = mysql_query(connection, create);
    create = strtok(NULL, ";");
}
printf("Completed to create table ans insert tuple₩n");
```

20160530.txt 파일에는 테이블의 생성과 tuple의 삽입을 위한 내용이 적혀있다. 먼저 파일을 읽기모드로 저장하여 모든 파일의 크기를 측정하고 buffer에 모든 내용을 담아둔다. 그 이후 정상적으로 mysql에 접속하면 buffer에서 ';'을 기준으로 문장을 읽어들여 sql에 데이터를 저장한다.

```
Connection Succeed

** Create table and insert tuple **
Completed to create table ans insert tuple
```

2) Select type

```
while (flag) {

    printf("₩n₩n------- SELECT QUERY TYPES -------₩n₩n");
    printf("₩t1. TYPE 1₩n");
    printf("₩t2. TYPE 2₩n");
    printf("₩t3. TYPE 3₩n");
    printf("₩t4. TYPE 4₩n");
    printf("₩t5. TYPE 5₩n");
    printf("₩t6. TYPE 6₩n");
    printf("₩t7. TYPE 7₩n");
    printf("₩t0. QUIT₩n");
    printf("---------------------------------₩n");
    printf("Select Number : ");
    scanf("%d", &type);
    getchar();
    switch (type) {
    case 0:
        flag = 0;
        break;
    case 1:
```

```
------- 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
---------------------------------
Select Number : _
```

프로그램의 전반적인 형태는 while문과 switch문을 통해 실행된다. 원하는 숫자를 선택하고 1~7을 선택하면 원하는 쿼리문을 실행할 수 있으며, 0은 종료, 그 외 입력은 다시 입력하도록 구성하였다.

3) Delete and Drop

```
FILE* fp2 = fopen("20160530_2.txt", "r");
char* buffer2 = NULL;

fseek(fp2, 0, SEEK_END);
size = ftell(fp2);
buffer2 = (char*)malloc(size + 1);
memset(buffer2, 0, size + 1);
fseek(fp2, 0, SEEK_SET);
fread(buffer2, size, 1, fp2);


const char* del = strtok(buffer2, ";");
while (del != NULL) {
    state = 0;
    state = mysql_query(connection, del);
    del = strtok(NULL, ";");
}
printf("₩n**  Completed to delete tuple and drop table. **₩n");
printf("Exit Program₩n");
```

```
---------------------------------
Select Number : 0

**  Completed to delete tuple and drop table. **
Exit Program
```

20160530_2.txt에는 tuple의 삭제와 테이블의 drop을 위한 내용이 적혀있다. Create and insertion과 마찬가지로 buffer에 내용을 저장한 후에 ';'를 기준으로 읽어들여 mysql안에 있는 데이터를 제거한다.

4.2 Queries

1) Type 1

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

** Tracking number X is reported to have been destroyed in an accident. **
Enter Tracking number X : 30001
Customer : 20160001, Name : Eamon, Contact : 01089234631
```

추적 번호를 입력하면 추적 번호에 해당하는 고객의 정보를 출력한다.

MySQL query

select customer_ID, customer_name, contract

from customer where customer_ID =

(select customer_Id from order_info natural join shipment where shipment_Id = Traking number X)

주문정보와 배송정보를 담고 있는 order_info와 shipment를 join하여 customer_ID를 도출한다. 이후에 해당 ID를 customer 테이블에서 찾아 필요한 정보를 출력한다.

```
printf("---- TYPE 1 ----₩n₩n");
printf("** Tracking number X is reported to have been destroyed in an accident. **₩n");

char Tnum[10];
printf("Enter Tracking number X : ");
scanf("%s", Tnum);
if (Tnum[0] == '0') break;


memset(query, 0, sizeof(query));
sprintf(query, "select customer_ID, customer_name, contract from customer where customer_ID = (se
state = 0;
state = mysql_query(connection, query);
if (state == 0)
{
    sql_result = mysql_store_result(connection);
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("Customer : %s, Name : %s, Contact : %s₩n", sql_row[0], sql_row[1], sql_row[2]);
        customer_ID = sql_row[0];
    }
    mysql_free_result(sql_result);
}
printf("₩n₩n");
```

1-1) Type 1-1

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

** Create a new shipment of replacement items. **
Shipment_ID : 30100, order_ID : 10000100
Product_ID Quantity
       101        1


** Completed to create a new shipment. **
```

Type-1의 서브 타입에서는 파손된 배송에 대한 상품을 다시 구성하여 배송을 시작한다. 새로운 주문 ID와 배송 ID를 구성하고 상품 정보를 받아와 내용을 출력한다.

MySQL query

`select product_ID, quantity from comprise where order_ID =`
`(select order_ID from order_info natural join shipment where shipment_Id = Traking number X)`
주문정보, 배송정보를 담은 테이블을 join하여 추적정보와 일치하는 order_ID를 선택한다. 이후 상품의 개수를 담고 있는 comprise에서 order_ID와 일치하는 상품의 ID와 개수를 결과에 담는다.

`insert into order_info values ('new_order', 'customer_ID', 'year-month-day')`
새로운 order_ID와 이전에 찾아 놓은 customer_ID, 오늘 날짜를 order_info 테이블에 삽입한다.

`insert into shipment values ('new_shipment', 'new_order', 'shipment_name', 'year-month-day', null)`
새로운 shipment_ID와 새로운 order_ID와 관련 정보를 shipment 테이블에 삽입힌다.

`insert into comprise values ('new_order', 'product_ID', 'quantity')`
담아둔 상품의 ID와 개수를 결과를 comprise 테이블에 삽입한다.

```
printf("₩n---- TYPE 1-1 ----₩n₩n");
printf("** Create a new shipment of replacement items. **₩n");

memset(query, 0, sizeof(query));
sprintf(query, "select product_ID, quantity from comprise where order_ID = (select order_ID from order_info natu
state = 0;
state = mysql_query(connection, query);
if (state == 0)
{
    sql_result = mysql_store_result(connection);
    printf("Shipment_ID : %d, order_ID : %d", new_shipment, new_order);
    memset(query, 0, sizeof(query));
    sprintf(query, "insert into order_info values ('%d', '%s', '%d-%d-%d');", new_order, customer_ID, t->tm_yea
    state = 0;
    state = mysql_query(connection, query);

    memset(query, 0, sizeof(query));
    sprintf(query, "insert into shipment values ('%d', '%d', 'shipment1', '%d-%d-%d', null);", new_shipment++,
    state = 0;
    state = mysql_query(connection, query);

    printf("₩nProduct_ID Quantity₩n");
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("%10s        %-5s₩n", sql_row[0], sql_row[1]);
        memset(query, 0, sizeof(query));
        sprintf(query, "insert into comprise values ('%d', '%s', '%s');", new_order, sql_row[0], sql_row[1]);
        state = 0;
        state = mysql_query(connection, query);
    }

    mysql_free_result(sql_result);
```

2) Type 2

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

** Find the customer who has bought the most (by price) in the past year. **
Enter year : 2021
CustomerID : 20160009, Name : Nonah, Total amount : 3600
```

해당 년도를 입력하면 금액 기준으로 그 해 가장 많이 구입한 고객의 ID와 이름, 구매 총액을 출력한다.

MySQL query

```
with tb as(select customer_ID, sum(quantity*product_price) as total_price
        from order_info natural join comprise natural join product
        where date(order_date) between 'year-01-01' and  'year-12-31'
        group by customer_ID order by total_price desc limit 1)
select customer_ID, customer_name, total_price from tb natural join customer
```

먼저 with를 이용하여 테이블을 구성한다. 주문정보(order_info)와 주문별 상품의 개수(comprise), 상품의 정보(product)를 담고 있는 테이블을 join하고 입력된 년도 사이의 값을 가져온다. 이때 customer_ID로 묶은 후 상품의 가격과 수량을 곱한 값을 더한 total_price를 도출하여 내림차순으로 정렬한 후 값을 하나만 취해 최대 값을 가져온다. 이후 이 테이블을 customer 테이블과 join하여 고객에 대한 정보를 출력한다.

```
printf("---- TYPE 2 ----₩n₩n");
printf("** Find the customer who has bought the most (by price) in the past year. **₩n");

char year[10];
char c_ID[10];
printf("Enter year : ");
scanf("%s", year);
if (year[0] == '0') break;

memset(query, 0, sizeof(query));
sprintf(query, "with tb as(select customer_ID, sum(quantity*product_price) as total_price from order_info na
state = 0;
state = mysql_query(connection, query);
if (state == 0)
{
    sql_result = mysql_store_result(connection);
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("CustomerID : %s, Name : %s, Total amount : %s₩n", sql_row[0], sql_row[1], sql_row[2]);
        sprintf(c_ID, sql_row[0]);
    }
    mysql_free_result(sql_result);
}
printf("₩n₩n");
```

2-1) Type 2-1

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

** Then find the product that the customer bought the most. **
Product ID: 113, Name : pakage13, Total amount : 2500
```

그 해의 가장 많이 구입한 상품의 ID, 이름, 총 금액을 출력한다.

MySQL query

```
select product_ID, product_name, quantity*product_price as total_price
from order_info natural join comprise natural join product
where date(order_date) between 'year-01-01' and  'year-12-31' and customer_ID = 'customer_ID'
group by product_ID order by total_price desc limit 1
```

주문정보(order_info)와 주문별 상품의 개수(comprise), 상품의 정보(product)를 담고 있는 테이블을 join하여 테이블을 구성하는데, 해당 년도 사이의 값을 가져오며, customer_ID가 최다 구매 고객과 일치하는 값을 찾는다.

```
printf("---- Subtypes in TYPE 2 ----₩n");
printf("₩t1. TYPE 2-1₩n");
printf("₩t0. QUIT₩n");
printf("--------------------------₩n");
printf("Select Number : ");
scanf("%d", &subtype);
if (subtype != 1) break;

printf("₩n---- TYPE 2-1 ----₩n₩n");
printf("** Then find the product that the customer bought the most. **₩n");
memset(query, 0, sizeof(query));
sprintf(query, "select product_ID, product_name, quantity*product_price as total_price from order_info na
state = 0;
state = mysql_query(connection, query);
if (state == 0)
{
    sql_result = mysql_store_result(connection);
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("Product ID: %s, Name : %s, Total amount : %s₩n", sql_row[0], sql_row[1], sql_row[2]);
    }
    mysql_free_result(sql_result);
}
printf("₩n₩n");
```

3) Type 3

```
**  Find all products sold in the past year. **
Enter year : 2021

Product_ID      Product_name
       001            Iphone
       003          GaluxyS
       004       Lgwashtower
       006       Lgwashtower
       007          Dvaccum
       008        Hpprinter
       009      LenovoLaptop
       010       Hwaweiphone
       011           Nophone
       012       AususLaptop
       013       MSIkeyboard
       101           pakage1
       102           pakage2
       105           pakage5
       106           pakage6
       109           pakage9
       112          pakage12
       113          pakage13
```

입력된 년도에 판매된 모든 상품의 ID와 상품명을 출력한다.

MySQL query

```
select distinct (product_ID), product_name
from order_info natural join comprise natural join product
where date(order_date) between 'year-01-01' and 'year-12-31'
```

주문정보와 주문된 상품을 담고 있는 테이블을 join하고 이후에 상품의 정보를 담고 있는 테이블을 join한다. 입력된 년도에 해당하는 tuple 만을 가져오는데, 중복된 상품을 제거하기 위해 distinct 를 이용하였다.

```
printf("---- TYPE 3 ----₩n₩n");
printf("**  Find all products sold in the past year. **₩n");

printf("Enter year : ");
scanf("%s", year);
if (year[0] == '0') break;

memset(query, 0, sizeof(query));
sprintf(query, "select distinct (product_ID), product_name from or

state = 0;
state = mysql_query(connection, query);
if (state == 0)
{
    printf("₩nProduct_ID      Product_name₩n");
    sql_result = mysql_store_result(connection);
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("%10s      %12s₩n", sql_row[0], sql_row[1]);
    }
    mysql_free_result(sql_result);
```

3-1) Type 3-1

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

**  Then find the top k products by dollar-amount sold.  **
 Which K? : 5

No. Product_ID  Product_name  Total amount
  1        011       Nophone         2880
  2        008     Hpprinter         2680
  3        006   Lgwashtower         2580
  4        113      pakage13         2500
  5        112      pakage12         2200
```

K를 입력 받아 금액 기준으로 해당 년도에 가장 많이 팔린 상품의 정보를 k개 만큼 출력한다.

MySQL query

```
select product_ID, sum(quantity*product_price) as total_price
from order_info natural join comprise natural join product
where date(order_date) between 'year-01-01' and 'year-12-31'
group by product_ID order by total_price desc limit K
```

주문정보(order_info)와 주문된 상품(comprise) join 하고 이후에 상품의 정보(product) join 한 후 해당 년도에 해당하는 tuple 을 가져온다. 이후에 상품의 ID 로 묶은 후에 상품의 수량과 가격을 곱한 총 금액 순으로 내림차순 정렬한 후에 K 만큼의 개수를 보여준다.

```
printf("---- Subtypes in TYPE 3 ----₩n");
printf("₩t1. TYPE 3-1₩n");
printf("₩t2. TYPE 3-2₩n");
printf("₩t0. QUIT₩n");
printf("----------------------------₩n");
printf("Select Number : ");
scanf(" %d", &subtype);
if (subtype == 0) break;
if (subtype == 1) {
    printf("₩n₩n---- TYPE 3-1 ----₩n₩n");
    printf("**  Then find the top k products by dollar-amount sold.  **₩n");
    printf(" Which K? : ");
    scanf(" %d", &k);
    getchar();
    memset(query, 0, sizeof(query));
    sprintf(query, "select product_ID, sum(quantity*product_price) as total_pr

    state = 0;
    state = mysql_query(connection, query);
    int num = 1;
    if (state == 0)
    {
        printf("₩nNo.  Product_name  Total amount₩n");
        sql_result = mysql_store_result(connection);
        while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
        {
            printf("%3d %12s  %12s₩n", num++, sql_row[0], sql_row[1]);
        }
        mysql_free_result(sql_result);
    }
```

3-2) Type 3-2

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

**   And then find the top 10% products by dollar-amount sold. **

No. Product_ID  Product_name  Total amount
  1        011       Nophone         2880
  2        008     Hpprinter         2680
```

해당 년도에 판매된 총 금액의 상위 10%안에 드는 상품의 정보를 출력한다.

MySQL query

```
with tb as(select product_ID, product_name, sum(quantity*product_price) as total_price
        from order_info natural join comprise natural join product
        where date(order_date) between 'year-01-01' and 'year-12-31'
        group by product_ID order by total_price desc)
select * from(select product_ID, product_name, total_price,
percent_rank() over (order by total_price desc)*100 as per from tb) a
where per <=10
```

먼저 with을 이용하여 테이블을 구성한다. 테이블을 만드는 과정은 Type-3에서 사용한 쿼리와 동일하다. 이후에 total price를 내림차순으로 구성한 테이블에서 percent_rank()를 활용하여 총 금액의 상위 퍼센트를 가져오고 그 값에 100을 곱하여 100%비율로 나타낸다. where에서 그 값이 10보다 작은 값을 가진 tuple만을 가져와 상품의 정보를 출력한다.

```
printf("₩n₩n---- TYPE 3-2 ----₩n₩n");
printf("**   And then find the top 10%% products by dollar-amount sold. **₩n");
memset(query, 0, sizeof(query));
sprintf(query, "with tb as(select product_ID, product_name, sum(quantity*product_price) a

state = 0;
state = mysql_query(connection, query);
int num = 1;
if (state == 0)
{
    printf("₩nNo. Product_ID  Product_name  Total amount₩n");
    sql_result = mysql_store_result(connection);
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("%3d %10s  %12s  %12s₩n", num++, sql_row[0], sql_row[1], sql_row[2]);
    }
    mysql_free_result(sql_result);
}
```

4) Type 4

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

**  Find all products by unit sales in the past year. **
Enter year : 2021

Product_ID  Product_name  Quantity
       001        Iphone         2
       003       GaluxyS         2
       004   Lgwashtower         1
       006   Lgwashtower         3
       007       Dvaccum         2
       008     Hpprinter         4
       009  LenovoLaptop         2
       010   Hwaweiphone         2
       011       Nophone         3
       012   AususLaptop         2
       013   MSIkeyboard         3
       101       pakage1         1
       102       pakage2         1
       105       pakage5         1
       106       pakage6         1
       109       pakage9         1
       112      pakage12         2
       113      pakage13         2
```

입력된 년도에 판매된 모든 상품의 ID와 상품명, 판매 수량을 출력한다.

MySQL query

```
select product_ID, product_name, sum(quantity)
from order_info natural join comprise natural join product
where date(order_date) between 'year-01-01' and 'year-12-31'
group by product_ID
```

주문정보와 주문된 상품을 담고 있는 테이블을 join하고 이후에 상품의 정보를 담고 있는 테이블을 join한다. 입력된 년도에 해당하는 tuple만을 가져오는데, product_ID를 그룹으로 묶어 중복된 상품의 값을 제거하면서 수량의 합계를 가져온다.

```
printf("---- TYPE 4 ----₩n₩n");
printf("**  Find all products by unit sales in the past year. **₩n");

printf("Enter year : ");
scanf("%s", year);
if (year[0] == '0') break;

memset(query, 0, sizeof(query));
sprintf(query, "select product_ID, product_name, sum(quantity) from order_info natural j
state = 0;
state = mysql_query(connection, query);

if (state == 0)
{
    printf("₩nProduct_ID  Product_name  Quantity₩n");
    sql_result = mysql_store_result(connection);
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("%10s  %12s  %8s₩n", sql_row[0], sql_row[1], sql_row[2]);
    }
    mysql_free_result(sql_result);
}
```

4-1) Type 4-1

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

**  Then find the top k products by unit sales.  **
 Which K? : 5

No. Product_ID  Product_name  Quantity
  1        008     Hpprinter         4
  2        011       Nophone         3
  3        013   MSIkeyboard         3
  4        006   Lgwashtower         3
  5        007       Dvaccum         2
```

K를 입력 받아 개수 기준으로 해당 년도에 가장 많이 팔린 상품의 정보를 k개 만큼 출력한다.

MySQL query

```
select product_ID, product_name, sum(quantity) as total_q
from order_info natural join comprise natural join product
where date(order_date) between 'year-01-01' and 'year-12-31'
group by product_ID order by total_q desc limit k
```

주문정보(order_info)와 주문된 상품(comprise) join 하고 이후에 상품의 정보(product) join 한 후 해당 년도에 해당하는 tuple 을 가져온다. 이후에 상품의 ID 로 묶은 후에 상품의 수량의 총 합을 기준으로 내림차순 정렬한 후에 K 만큼의 개수를 보여준다.

```
printf("₩n₩n---- TYPE 4-1 ----₩n₩n");
printf("**  Then find the top k products by unit sales.  **₩n");
printf(" Which K? : ");
scanf(" %d", &k);
getchar();
memset(query, 0, sizeof(query));
sprintf(query, "select product_ID, product_name, sum(quantity) as total_q from order_ir

state = 0;
state = mysql_query(connection, query);
int num = 1;
if (state == 0)
{
    printf("₩nNo. Product_ID  Product_name  Quantity₩n");
    sql_result = mysql_store_result(connection);
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("%3d %10s  %12s  %8s₩n", num++, sql_row[0], sql_row[1], sql_row[2]);
    }
    mysql_free_result(sql_result);
}
```

4-2) Type 4-2

```
**    And then find the top 10% products by unit sales. **

No. Product_ID  Product_name  Quantity
  1        008     Hpprinter         4
  2        006   Lgwashtower         3
  3        011       Nophone         3
  4        013   MSIkeyboard         3
```

해당 년도에 판매된 총 판매 개수의 상위 10%안에 드는 상품의 정보를 출력한다.

MySQL query

```
with tb as (select product_ID,  product_name, sum(quantity) as total_quantity
        from order_info natural join comprise natural join product
        where date(order_date) between '%s-01-01' and '%s-12-31'
        group by product_ID order by total_quantity desc)
select * from(select product_ID,  product_name, total_quantity,
percent_rank() over (order by total_quantity desc)*100 as per from tb) a
where per <= 10
```

먼저 with을 이용하여 테이블을 구성한다. 테이블을 만드는 과정은 Type-4에서 사용한 쿼리와 동일하다. 이후에 total_quantity를 내림차순으로 구성한 테이블에서 percent_rank()를 활용하여 총 개수의 상위 퍼센트를 가져오고 그 값에 100을 곱하여 100%비율로 나타낸다. where에서 그 값이 10보다 작은 값을 가진 tuple만을 가져와 상품의 정보를 출력한다.

```
printf("₩n₩n---- TYPE 4-2 ----₩n₩n");
printf("**    And then find the top 10% products by unit sales. **₩n");
memset(query, 0, sizeof(query));
sprintf(query, "with tb as (select product_ID,  product_name, sum(quantity) as total_c

state = 0;
state = mysql_query(connection, query);
int num = 1;
if (state == 0)
{
    printf("₩nNo. Product_ID  Product_name  Quantity₩n");
    sql_result = mysql_store_result(connection);
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("%3d %10s  %12s  %8s₩n", num++, sql_row[0], sql_row[1], sql_row[2]);
    }
    mysql_free_result(sql_result);
}
```

5) Type 5

```
---- TYPE 5 ----

**  Find those products that are out-of-stock at every store in California. **
Enter location of store : California
Product_ID  Product_name
       004   Lgwashtower
       006   Lgwashtower
       010   Hwaweiphone
```

원하는 창고의 위치를 입력하고 해당 창고에서 품절된 상품을 검색하여 결과를 출력한다.

MySQL query

```
select product_ID, product_name
from store natural join warehouse natural join product
where inventory = 0 and warehouse_location = 'California'
```

창고(warehouse)와 상품의 정보(product), 그리고 상품의 재고가 저장되어 있는 테이블(store)을 각각 join한다. 이후에 재고가 0이면서 창고의 위치가 받아온 위치인 tuple의 product_ID와 product_name을 출력한다.

```
    printf("---- TYPE 5 ----₩n₩n");
    printf("**  Find those products that are out-of-stock at every store in California. **₩n");

    char location[10];
    printf("Enter location of store : ");
    scanf("%s", location);
    if (location[0] == '0') break;

    memset(query, 0, sizeof(query));
    sprintf(query, "select product_ID, product_name from store natural join warehouse natural jo
    state = 0;
    state = mysql_query(connection, query);


    if (state == 0)
    {
        printf("₩Product_ID  Product_name₩n");
        sql_result = mysql_store_result(connection);
        while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
        {
            printf("%10s  %12s₩n", sql_row[0], sql_row[1]);
        }
        mysql_free_result(sql_result);
    }
```

6) Type 6

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

**  Find those packages that were not delivered within the promised time. **

Product_ID  Products(Pakage)  Expected date  Completion date
       104           pakage4     2022-02-20       2022-02-22
       107           pakage7     2022-02-11       2022-02-12
       110          pakage10     2022-05-29       2022-05-30
```

약속된 시간 안에 배달되지 않은 패키지의 정보와 예정 배송일, 배송 완료일을 출력한다.

MySQL query

```
select product_ID, product_name, shipment_date, completion_date
from order_info natural join shipment natural join comprise natural join product
where product_ID > 100 and date(shipment_date) < date(completion_date)
```

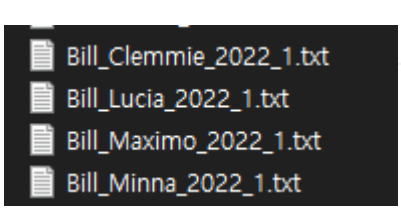
주문정보(order_info)와 배송정보(shipment)를 join하고 주문 상품(comprise)를 join한 이후 상품(product)을 join한다. 이후에 product_ID가 100이상(일반 상품은 ID가 100미만의 값을 갖고, 패키지 상품은 100이상의 값을 가짐)인 값과, 배송 완료일이 예정 배송일 보다 큰 값을 찾아 출력한다.

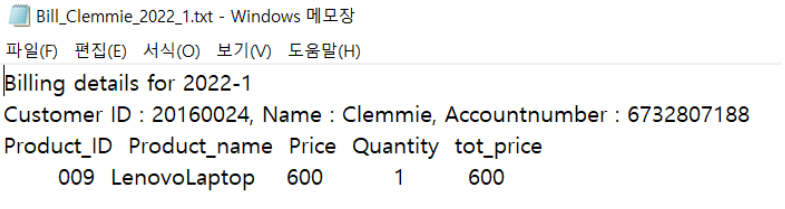
```
printf("---- TYPE 6 ----₩n₩n");
printf("**  Find those packages that were not delivered within the promised time. **₩n₩n");


memset(query, 0, sizeof(query));
sprintf(query, "select product_ID, product_name, shipment_date, completion_date from order_info na
state = 0;
state = mysql_query(connection, query);

if (state == 0)
{
    printf("₩Product_ID  Products(Pakage)  Expected date  Completion date  ₩n");
    sql_result = mysql_store_result(connection);
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("%10s  %16s  %13s  %15s₩n", sql_row[0],sql_row[1], sql_row[2], sql_row[3]);
    }
    mysql_free_result(sql_result);
}
```

7) Type 7

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

**  Generate the bill for each customer for the past month. **
Enter month : 1

Generate bill for the following customer_ID list

20160017  20160020  20160024  20160029

**  Completed to generate bill. **
```

해당 월에 대한 정보를 입력하면 해당 월에 거래 내역이 있는 contractor에 대한 bill을 생성한다.





MySQL query

```
select distinct customer_ID, customer_name, accountnumber
from order_info natural join contractor natural join customer
where date(order_date) between 'year-month-01' and 'year-month-31
```

주문 정보(order_info)와 계약자(contractor)를 join하고 고객(customer)을 join한다. 이후에 해당 년도와 해당 입력한 월에 거래된 내역이 있는 customer_ID, name, accountnumber을 가져오는데 distinct를 이용하여 중복된 값이 제거된 tuple이 결과에 저장한다.

```
select product_ID, product_name, product_price, quantity, (product_price*quantity) as tot_price
from order_info natural join contractor natural join comprise
natural join product natural join customer
where date(order_date) between 'year-month-01' and 'year-month-31' and customer_ID = 'customer_ID'
```

주문 정보(order_info)와 계약자(contractor)를 join하고 주문 상품(comprise)에 join, 해당 년, 월 사이의 값을 가져온다. 그리고 customer_ID가 앞선 쿼리에서 저장된 customer_ID와 일치하는 tuple을 가져와 저장되어 있는 product_ID, product_name, price, quantity, price와 quantity를 곱한 tot_price를 가져와 bill안에 저장한다.

```c
printf("---- TYPE 7 ----₩n₩n");
printf("**  Generate the bill for each customer for the past month. **₩n");

FILE* op;
char bill[100]; char month[10];
printf("Enter month : ");
scanf("%s", month);
if (month[0] == '0') break;

memset(query, 0, sizeof(query));
sprintf(query, "select distinct customer_ID, customer_name, accountnumber from order_info
state = 0;
state = mysql_query(connection, query);
if (state == 0)
{
    printf("₩nGenerate bill for the following customer_ID list₩n₩n");
    sql_result = mysql_store_result(connection);
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("%s  ", sql_row[0]);
        memset(query, 0, sizeof(query));
        sprintf(query, "select product_ID, product_name, product_price, quantity, (product
        state = 0;
        state = mysql_query(connection, query);
        if (state == 0)
        {
            sql_result2 = mysql_store_result(connection);
            while ((sql_row2 = mysql_fetch_row(sql_result2)) != NULL)
            {
                sprintf(bill, "Bill_%s_2022_%s.txt", sql_row[1], month);
                op = fopen(bill, "w");
                fprintf(op, "Billing details for last month₩nCustomer ID : %s, Name : %s,
                fprintf(op, "Product_ID  Product_name  Price  Quantity  tot_price₩n");
                fprintf(op, "%10s  %12s  %5s  %8s  %9s₩n", sql_row2[0], sql_row2[1], sql_
                fclose(op);
            }
            mysql_free_result(sql_result2);
        }
    }
    mysql_free_result(sql_result);
    printf("₩n₩n**  Completed to generate bill. **₩n");
}
```

Query1에서는 해당 월에 거래내역이 있는 contractor의 정보를 가져와 sql_result에 저장하고 query2에서는 sql_result에 저장된 row의 값을 하나씩 불러오면서 customer_ID에 알맞은 상품과 가격을 sql_result2에 저장하여 값을 차례대로 bill에 저장한다.

Logical Schema Diagram