DB Project 2

20191164 최윤지

1. 프로젝트 개요

Project1 에서 결계한 Database 를 BCNF 에 맞도록 정규화하여 데이터를 효율적으로 이용할 수 있는 database 를 설계한다. 또한 C++ 프로그램을 이용한 ODBC 와 MySQL DBMS 를 이용하여 query 를 검색이 가능한 프로그램을 완성한다.

2. BCNF Normalization

Project1 에서 작성한 Logical schema 가 BCNF dependency 을 만족하는지를 확인하기 위해서는 각 relation 과 entity 의 Functional Dependency 를 확인해야한다. 강의 시간 중 배운 process 와 동일하게 BCNF simplified test 를 이용하였다. BCNF simplified test 에서는 FD 가 trival 하거나 ($\beta \subseteq \alpha$) α 가 슈퍼키여야 한다. (이후 query 를 작성하는 과정에서 Project1 과 다르게 수정한 부분이 있으나 이것 역시 BCNF dependency 를 만족하는지 확인하였다.)

확인한 결과, 기존 logical schema 는 BCNF 를 모두 만족하여 decomposition 이 필요한 부분은 없었다. 아래에 그 과정과 각 relation 과 entity 의 FD 를 서술하였다.

(1) Package(<u>package_id</u>, type, weight, teimliness, content, value, service_type, charge, customer_id, shipment_id)

FD 는 다음과 같다.

- Package_id -> type, weight, content, value, service_type, charge, customer_id
- Customer_id -> name, number, address, last_year_frequency, last_year_total_charge, shipper_id
- Shipper_id -> name, number, account_number
- Shipment_id -> recipient_name, recipient_number, recipient_number, depart_time, promised_time, delivered_time

좌측이 전부 슈퍼키에 해당한다.

(2) Customer(customer_id, name, number, address, last_year_frequency, last_year_total_charge, shipper_id)

FD 는 다음과 같다.

- Customer_id -> name, number, address, last_year_frequency, last_year_total_charge, shipper_id
- Shipper_id -> name, number account_number

좌측이 전부 슈퍼키에 해당한다.

(3)Shipper(Shipper_id, name, number, account_number)

FD 는 다음과 같다.

- Sihpper_id -> name, number, account_number

PK 인 shipper_id 외에 다른 속성 간의 FD 가 존재하지 않는다.

(4)Bill(bill_id, package_id, payment_method, bill_type, issued_date, total_charge, customer_id)

FD 는 다음과 같다.

- Bill_id -> package_id, payment_method, bill_type, issued_date, total_charge, customer_id
- Package_id -> type, weight, content, value, service_type, charge, customer_id
- Customer_id -> name, number, address, last_year_frequency, last_year_total_charge, shipper_id
- Shipper_id -> name, number, account_number

좌측이 전부 슈퍼키에 해당한다.

(5)Delivered(package_id, dropped_time, check_prepaid, shipment_id)

FD 는 다음과 같다.

- Package_id -> dropped_time, check_prepaid, shipment_id, type, weight, content, value, service_type, charge, customer_id
- Customer_id -> name, number, address, last_year_frequency, last_year_total_charge, shipper_id
- Shipper_id -> name, number, account_number
- Shipment_id -> recipient_name, recipient_number, recipient_number, depart_time, promised_time, delivered_time

좌측이 전부 슈퍼키에 해당한다.

(6)Shipment_id(recipient_name, recipient_number, recipient_number, depart_time, promised_time, delivered_time)

FD 는 다음과 같다.

- Shipment_id -> recipient_name, recipient_number, recipient_number, depart_time, promised_time, delivered_time

PK 인 Shipment_id 외에 다른 속성간의 FD 가 존재하지 않는다.

(7) Tracking(tracking_id, status, tracked_time, shipment_id, vehicle_type, serial_number, address)

FD 는 다음과 같다.

- tracking_id -> status, tracked_time, shipment_id, vehicle_type, serial_number, address
- Shipment_id -> recipient_name, recipient_number, recipient_number, depart_time, promised_time, delivered_time
- Vehicle_type, serial_number-> driver, driver_number
- Address -> type, name, number

좌측이 전부 슈퍼키에 해당한다.

(8) Vehicle(Vehicle_type, serial_number, driver, driver_number)

FD 는 다음과 같다.

- Vehicle_type, serial_number -> driver, driver_number

PK 인 Vehicle_type, serial_number 외에 다른 속성 간의 FD 가 존재하지 않는다.

(9) Location_info (address, type, name, number)

FD 는 다음과 같다.

- Address -> type, name, number

PK 인 address 외에 다른 속성 간의 FD 가 존재하지 않는다.

(10) last_visit(address, last_visited_address)

FD 는 다음과 같다.

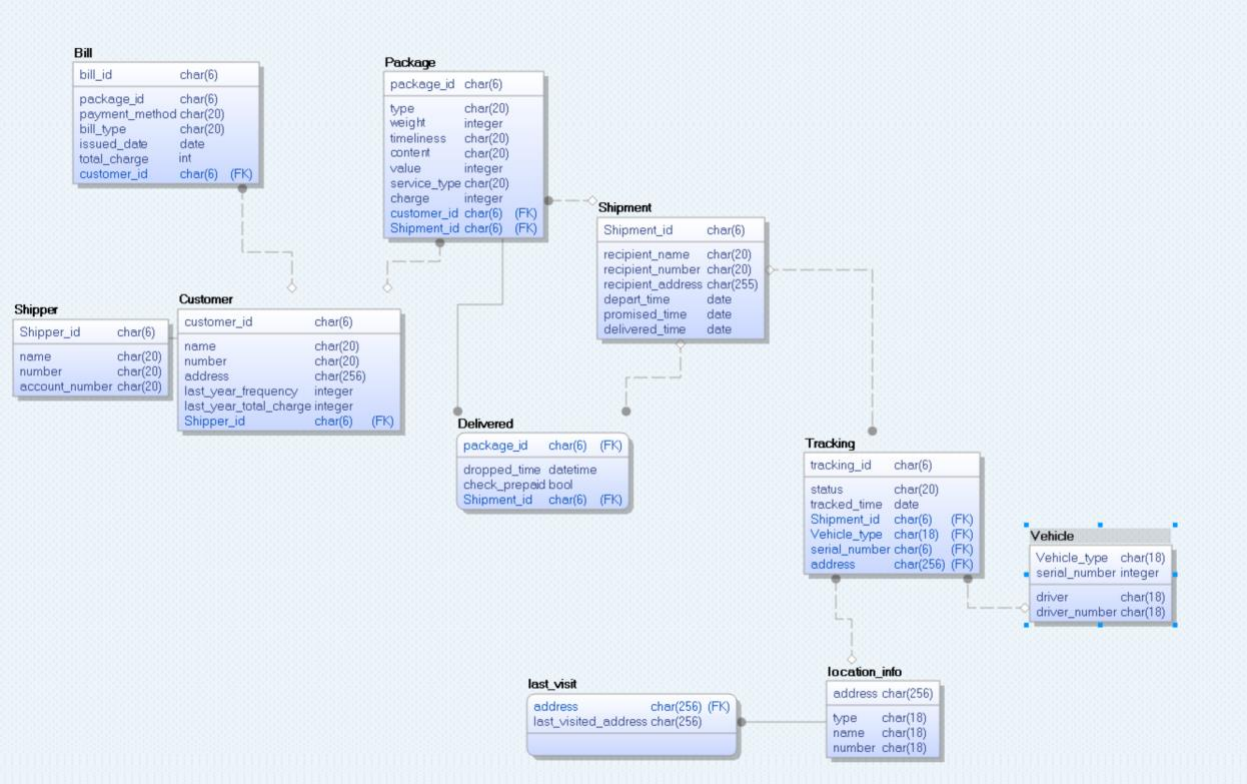
- Address -> type, name, number

PK 인 address 외에 다른 속성 간의 FD 가 존재하지 않는다.

따라서 위의 결과를 바탕으로 project1 과 동일한 Logical schema 를 사용하였다.

3. Physical Schema Diagram

2.에서의 결과를 바탕으로 ERWin 프로그램에서 physical schema diagram 을 작성하였다.



각각의 data type 과 domain, constraints, relationship type, null 값 허용 여부 등은 다음과 같다.

(1) Shipper

shipper_id - varchar(6): 'SXXXXX' 형식으로 저장하여 동명이인 등으로 인한 혼선을 방지하였다. PK 이므로 null 이 될 수 없다.

name - varchar(20): shipper 의 이름이다.

number varchar(20): shipper 의 전화번호이다. "010-1234-4567" 형식으로 저장된다.

account_number varchar(30): shipper 의 계좌번호이다. "000-000000-00000"과 같은 형식으로 저장된다.

PK 를 포함하여 전부 필수적인 정보이므로 null 이 될 수 없도록 설정하였다.

(2)shipment

shipment_id - varchar(6): "SPXXXX" 형식으로 저장한다. PK 이므로 null 이 될 수 없다.

recipient_name - varchar(20) not null,

recipient_number - varchar(20) not null,

recipient_address - varchar(255) not null,

: 수신자의 이름, 전화번호, 주소이다. 전부 문자열로 저장하였으며 address 는 그 길이를 고려하여 크기를 255 로 지정해주었다. 영문으로 저장되며, 설정한 테스트 데이터는 랜덤하게 지정된 미국의 주소를 사용하여 테스트했다. 전부 null 값이 될 수 없다.

depart_time - DATE not null
promised_time - DATE not null
delivered_time - DATE not null

: 출발한 시간, 도착 예정 시간, 실제 도착 시간이다. “2022-01-01”과 같은 형식의 DATE 타입으로 저장된다. 전부 null 값이 될 수 없다.

(3)customer

customer_id - varchar(6): “CXXXXX” 형식으로 저장한다. PK 이므로 null 이 될 수 없다.
Name - varchar(20) not null,
Address - varchar(255) not null,

: 고객의 이름과 주소이므로 null 값이 될 수 없고 문자열로 저장하였다.

last_year_frequency - int not null,
last_year_total_charge - int not null,

작년에 해당 고객이 이용한 빈도수와 총 구매액이다. Int 형을 사용하여 숫자로 저장하였으며 각각의 단위는 회/원으로 고려하였다. 둘 다 Null 값이 될 수 없다.

shipper_id - varchar(6): 거래하는 shipper 가 존재하는 경우 shipper 의 id 를 저장하도록 하였다. Shipper 와 거래하지 않는 고객도 존재하므로 null 값이 될 수 있으며 shipper 과 many to one 의 관계를 갖고 foreign key constraint 가 적용된다. 다만 null 값이 가능하므로 shipper 가 삭제되는 경우 null 로 값을 저장하도록 하였다.

(4) package

package_id - varchar(6) : “CXXXXX” 형식으로 저장한다. PK 이므로 null 이 될 수 없다.

type - varchar(20): package 의 type 으로, 명세서에서 주어진 바와 같이 “flat envelope”, “small box”, “large box” 등의 패키지 타입을 저장한다. 따라서 문자열로 저장하였으며 null 값이 될 수 없다.

weight – int: package 의 무게이다. Kg 단위로 생각하여 설정하였으며 int 형 자료형을 사용하였고 역시 null 값이 될 수 없다.

timeliness - varchar(20): 명세서에서 주어진 바와 같이 배달의 limliness 를 “overnight”, “second day”, “longer”로 저장하였다. 문자열이 되어야 하며 null 값이 될 수 없다.

content - varchar(20): 패키지의 내용물이며, 위험물인지 등을 파악할 수 있다. 문자열로 저장되며 null 값이 될 수 없다.

value – int: 패키지 내용물의 가격이다. Null 값이 될 수 없으며 정수형으로 저장된다.

service_type - varchar(30): 위의 package type, weight, timeliness 등을 종합적으로 고려하여 분류하는 service type 을 설정하였다. 임의로 “A”, “B” 와 같은 알파벳으로 저장하였으며, null 값이 될 수 없다. 만일 복합적으로 고려해야하는 요소가 적다면 해당 항목은 사용하지 않을 수도 있다.
charge – int: 패킹하는데 필요한 비용으로, value 와는 다른 값이다. Int 형을 사용하였고 null 값이 될 수 없다.
customer_id - varchar(6): 해당 패키지를 주문한 customer 의 Id 이다.
shipment_id varchar(6): 해당 패키지가 배송되는 shipment 의 id 다.
Customer_id 와 shipment_id 는 각각 customer, shipment 속성을 reference 하는 FK 이며 many to one 의 관계가 성립한다. foreign key constraint 를 만족하도록 설정하였다. 또한 referenced 되는 테이블에서 각각의 속성이 삭제되면 null 값이 되도록 설정하였기 때문에 null 값을 허용하였다.

(5) bill

bill_id - varchar(6): “BXXXX’와 같은 형식으로 저장하는 문자열을 사용한다. PK 이므로 null 값이 될 수 없다.
package_id - varchar(6) not null: 해당 bill 의 package 이므로 null 값이 될 수 없고 package_id 를 가져와서 구분할 수 있도록 한다.
payment_method - varchar(20) not null: 해당 bill 이 어떤 method 로 결제되었는지 저장하는 항목이다. “Credit Card”, “Cash”와 같은 문자열로 저장되며 null 값이 될 수 없다.
bill_type - varchar(20) not null: service type 을 저장하도록 하였으며 null 값이 될 수 없다.
issued_date - DATE not null: Bill 이 발행된 날짜로 “2022-01-01”과 같은 형식의 DATE 타입을 사용하였다.
total_charge int not null: bill 에서 전체 가격을 계산하여 저장하도록 하였다. 따라서 정수형 자료형이며, null 값이 될 수 없다.
customer_id varchar(6): bill 의 주인이 되는 customer 을 저장하기 위한 customer_id 이다.
Customer_id 와 package id 는 각각 customer, package 속성을 reference 하는 FK 이며 many to many 의 관계가 성립한다. foreign key constraint 를 만족하도록 설정하였다. 또한 referenced 되는 테이블에서 각각의 속성이 삭제되면 null 값이 되도록 설정하였기 때문에 null 값을 허용하였다.

(6) Delivered

package_id varchar(6) – 어떤 package 에 관한 delivery 기록인지를 살피는 PK 이다.

dropped_time DATETIME – package 가 배달 완료된 시간을 알리는 DATETIME 형식으로, “2022-01-01 00:12:30”과 같은 형식으로 저장된다. Null 값이 될 수 없다.
check_prepaid Boolean: 미리 결제가 완료되었는지를 확인하기 위한 Boolean 자료형이다. Null 값이 될 수 없으며 T/F 로 구분된다.
shipment_id varchar(6) – 어떤 배달에 포함되어 배송되었는지를 확인하기 위한 shipment_id 이다.
package_id 와 shipment_id 는 각각 package, shipment 속성을 reference 하는 FK 이며 foreign key constraint 를 만족하도록 설정하였다. 또한 referenced 되는 테이블에서 각각의 속성이 삭제되면 연쇄적으로 삭제되도록 설정하였다.

(7) vehicle

vehicle_type varchar(20): truck, train 등의 운송기기 종류를 저장하는 문자열 정보이다.
serial_number varchar(4): 숫자 4 자리로 설정하였지만 덧셈 등의 연산을 사용하지는 않으므로 문자열을 사용하여 저장하였다.
Vehicle_type 과 serial_number 을 PK 로 하여 운송기기를 구분할 수 있도록 하였다. 각각의 id 로 구분을 하기에는 직관적이지 않고 id 에 따라 분류법을 다르게 하는 등의 추가적인 과정이 번거롭다고 판단하였다. 따라서 null 값이 될 수 있다.
driver varchar(20) not null,
driver_number varchar(20) not null,
문자열 타입을 이용하여 driver 의 이름과 번호를 저장한다. Null 값이 될 수 없다.

(8) location_info

laddress varchar(255): 주소로 address information 을 구분하도록 PK 로 설정하였다. 배송지가 될 수 있는 모든 주소를 id 화 하여 저장할 수 없다고 판단하였다. 또한 주소이므로 256 사이즈의 문자열로 저장하였다. Null 값이 될 수 없다.
type varchar(20) not null: 주소지의 타입을 저장하기 위한 항목이다. 허브, 창고, 일반 주소지 등으로 구분할 수 있도록 문자열로 저장하였다. Null 값이 될 수 없다.
name varchar(20), number varchar(20): 해당 주소지의 관리자와 연락처이다. Null 값이 될 수 없도록 설정하였으며 문자열을 이용하였다.

(9) tracking

tracking_id varchar(6): “TXXXXX” 형식으로 저장하는 PK 이다. Null 값이 될 수 없고 문자열로 저장하였다.
status varchar(50): 현재 상태를 저장하기 위한 항목이다. “preparing”, “done” “in process”와 같이 구분하도록 문자열로 지정하였으며 Null 값이 될 수 없다.

tracked_time: 조회를 시도한 날짜이다. “2022-01-01”과 같은 형식을 갖는 DATE type 을 사용하였고 null 값이 될 수 없다.
shipment_id varchar(6): 명세서 상에서 package 는 하나의 shipment 로 묶여서 배달된다고 하였으므로 어떠한 배송을 이용해 배달되고 있는지를 구분하기 위해 shipment_id 를 저장하였다. Shipment 없이 정보를 가질 순 없으므로 Null 값이 될 수 없다.

- shipment_id 는 shipment 를 reference 하는 FK 이므로 FK constraint 를 따른다. Null 값이 될 수 없으므로 Shipment 테이블에서 정보가 삭제되면 연쇄적으로 정보가 삭제된다. (delete on cascade)

vehicle_type varchar(20), serial_number varchar(4) : 현 시점에서 어떤 운송 수단을 이용하는지 저장하기 위한 항목이다.

- vehicle_type, serial_number 는 vehicle 를 reference 하는 FK 이므로 FK constraint 를 따른다. Null 값이 될 수 없으므로 vehicle 테이블에서 정보가 삭제되면 연쇄적으로 정보가 삭제된다. (delete on cascade)

laddress varchar(255): 현재 위치한 정보이다.

- laddress 는 address_info 를 reference 하는 FK 이므로 FK constraint 를 따른다. Null 값이 될 수 없으므로 address_info 테이블에서 정보가 삭제되면 연쇄적으로 정보가 삭제된다. (delete on cascade)

3. Queries

1)

ODBC 처리를 용이하게 하기 위해 두 개의 txt 파일을 사용하였다. 첫 번째는 table 의 생성과 data 의 insert 를 위한 파일이고, 두번째는 program exit 시 데이터를 삭제하고 table 을 drop 하기 위한 파일이다.

(1)

```
create table shipper(shipper_id varchar(6), name varchar(20) not null, number varchar(20) not null, account_number varchar(30) not null, primary key
(shipper_id))
create table shipment(shipment_id varchar(6), recipient_name varchar(20) not null, recipient_number varchar(20) not null, primary key(shipment_id))
alter table shipment add (recipient_address varchar(255) not null, depart_time DATE not null, promised_time DATE not null, delivered_time DATE not null
create table customer(customer_id varchar(6), name varchar(20) not null, address varchar(255) not null, last_year_frequency int not null, primary
key(customer_id))
alter table customer add ( last_year_total_charge int not null, shipper_id varchar(6))
alter table customer add constraint fk_shipper_id foreign key (shipper_id) references shipper (shipper_id) on delete set null
create table package(package_id varchar(6), type varchar(20) not null, weight int not null, timeliness varchar(20) not null, primary key(package_id))
alter table package add(content varchar(20) not null, value int not null, service_type varchar(30) not null, charge int not null, customer_id
varchar(6), shipment_id varchar(6))
alter table package add constraint fk_cus_id foreign key (customer_id) references customer (customer_id) on delete set null
alter table package add constraint fk_sh_id foreign key (shipment_id) references shipment (shipment_id) on delete set null
create table bill(bill_id varchar(6), package_id varchar(6) not null, payment_method varchar(20) not null, bill_type varchar(20) not null, primary
key(bill_id))
alter table bill add(issued_date DATE not null,  total_charge int not null, customer_id varchar(6))
alter table bill add constraint fk_cus_id foreign key (customer_id) references customer (customer_id) on delete set null
create table delivered(package_id varchar(6), dropped_time DATETIME, check_prepaid boolean, shipment_id varchar(6), primary key(package_id))
alter table delivered add constraint fk_pk_id foreign key (package_id) references package (package_id) on delete cascade
alter table delivered add constraint fk_sh_id foreign key (shipment_id) references shipment (shipment_id) on delete set null
create table vehicle (vehicle_type varchar(20), serial_number varchar(4), driver varchar(20) not null, driver_number varchar(20) not null, primary
key(vehicle_type, serial_number))
create table location_info(laddress varchar(255), type varchar(20) not null, name varchar(20) not null, number varchar(20) not null, primary key
(laddress))
create table last_visit(laddress varchar(255), last_visited_address varchar(255), primary key(laddress, last_visited_address))
alter table last_visit add constraint fk_li_ld foreign key (laddress) references location_info (laddress) on delete cascade
create table tracking(tracking_id varchar(6), status varchar(50) not null, tracked_time DATE not null, shipment_id varchar(6), primary key(tracking_id)
alter table tracking add (vehicle_type varchar(20), serial_number varchar(4), laddress varchar(255))
alter table tracking add constraint fk_sh_id foreign key (shipment_id) references shipment (shipment_id) on delete cascade
alter table tracking add constraint fk_vh_tynum foreign key (vehicle_type, serial_number) references vehicle (vehicle_type, serial_number) on delete
cascade
alter table tracking add constraint fk_li_add foreign key (laddress) references location_info (laddress) on delete set null
insert into shipper values ("S11111","Sophea Shani","010-1234-5678","123-456-123456")
insert into shipper values ("S11211","Gil Garnett","010-1234-5678","123-456-123456")
insert into shipper values ("S14411","Ravid Senna","010-1234-5678","123-456-123456")
insert into shipper values ("S15191","Lee Sohee","010-1234-5678","123-456-123456")
insert into shipper values ("S81011","Kim Min Jun","010-1234-5678","123-456-123456")
insert into shipment values("SP1233", "Song So her", "010-1111-1111", "Daejeon, See-gu, Duncan-dong", "2022-01-22", "2022-01-24", "2022-01-24")
insert into shipment values("SP1223", "Kim Hye Rin", "010-1411-1111", "2175 Quiet Valley Lane", "2022-01-22", "2022-01-24", "2022-01-25")
insert into shipment values("SP2123", "Kim Min Hee", "010-1411-1555", "Seoul, Mapo-gu, Duncan-dong", "2022-01-22", "2022-01-24", "2022-01-25")
insert into shipment values("SP4523", "Lee Hye Rin", "010-2341-1414", "2163 Smith Road", "2022-03-22", "2022-03-25", "2022-03-25")
insert into shipment values("SP1623", "Song kyung", "010-3515-2333", "4071 Maud Street", "2022-02-12", "2022-02-14", "2022-01-13")
insert into shipment values("SP1923", "Hye Ri NA", "010-5555-5555", "2081 Conaway Street", "2022-07-18", "2022-07-20", "2022-07-21")
insert into shipment values("SP2323", "Yu Mi Ran", "010-1123-5531", "2466 Blair Court", "2022-01-22", "2022-01-24", "2022-01-25")
insert into shipment values("SP7223", "David Rin", "010-1231-5124", "1113 Roguski Road", "2022-11-02", "2022-01-05", "2022-01-04")
insert into shipment values("SP8223", "Kim Hye Rin", "010-1245-6654", "2194 Taylor Street", "2022-02-12", "2022-02-14", "2022-02-15")
insert into customer values ("C11012", "Lee Min Ho", "Seoul, Sogangdaegil 16, Sinsu-dong", 5, 500000, "S11111")
insert into customer values ("C11200", "Kim Jun Ho", "Seoul, Sogangdaegil 16, Sinsu-dong", 7, 1500000, NULL)
```

(2)

```
drop table tracking
drop table last_visit
drop table location_info
drop table vehicle
drop table delivered
drop table bill
drop table package
drop table customer
drop table shipment
drop table shipper
```

2)

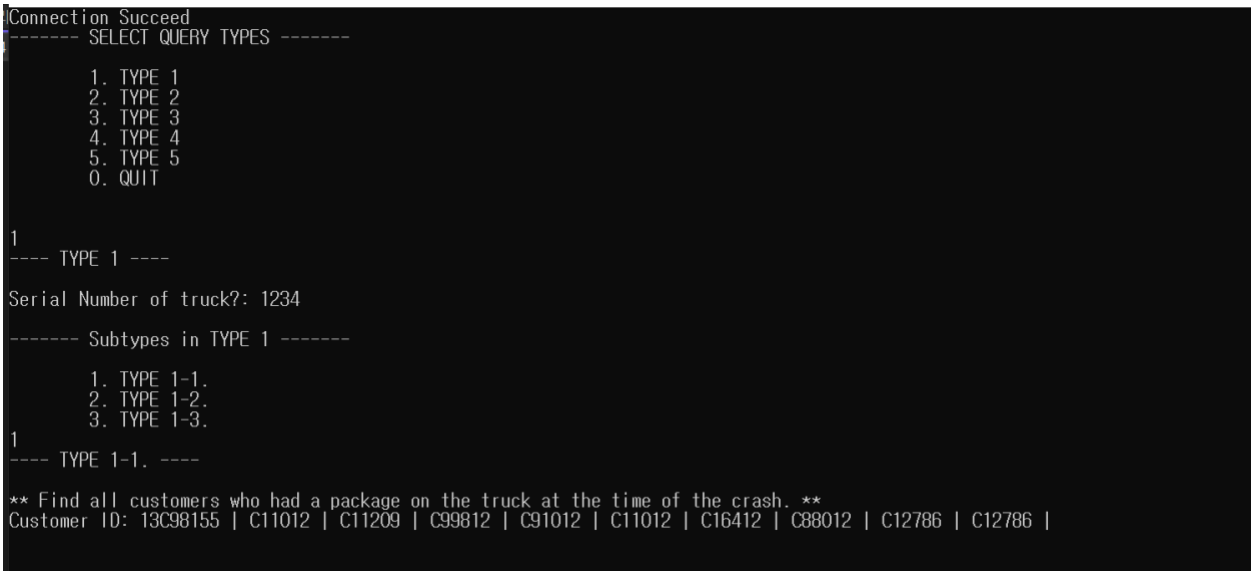
(1) TYPE 1

- subtype 1

Crush 된 트럭에서 마지막 tracked time 을 찾고 해당 shipment id 를 이용하여 package list 를 찾는다. 동명이인이 있을 수 있으므로 customer 의 id 를 출력하였다. 다음과 같은 query 를 사용하였다.

```
string query1_1 = "select P.customer_id from tracking T, shipment S, package P
where P.shipment_id = S.shipment_id and T.shipment_id = S.shipment_id and
tracked_time = (select MAX(tracked_time) from tracking) and vehicle_type = 'truck'
and serial_number = '";
 query1_1 = query1_1 + truck + "'";
```

실행한 결과는 다음과 같다.



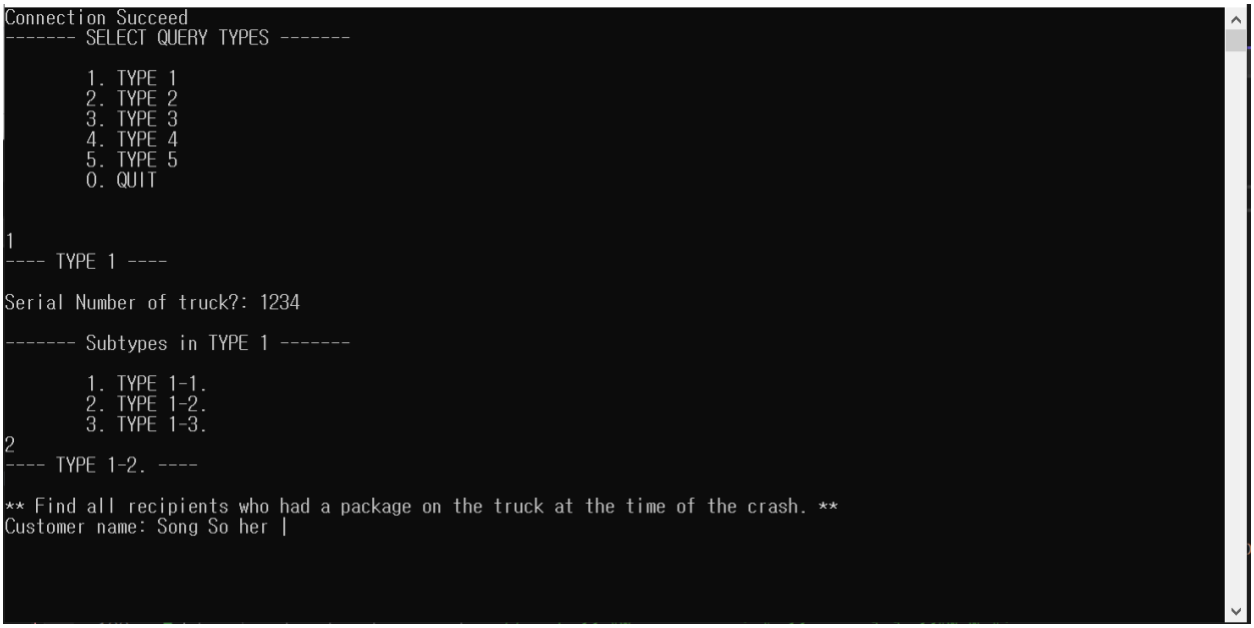
- subtype 2

Crush 된 트럭에서 마지막 tracked time 을 찾고 해당 shipment id 를 가지는 Recipient 정보를 찾았다.. 다음과 같은 query 를 사용하였다.

```
string query1_2 = "select S.recipient_name from tracking T, shipment S, package
P where P.shipment_id = S.shipment_id and T.shipment_id = S.shipment_id and
tracked_time = (select MAX(tracked_time) from tracking) and vehicle_type = 'truck'
and serial_number = '";
```

```
query1_2 = query1_2 + truck + "'";
```

실행한 결과는 다음과 같다.



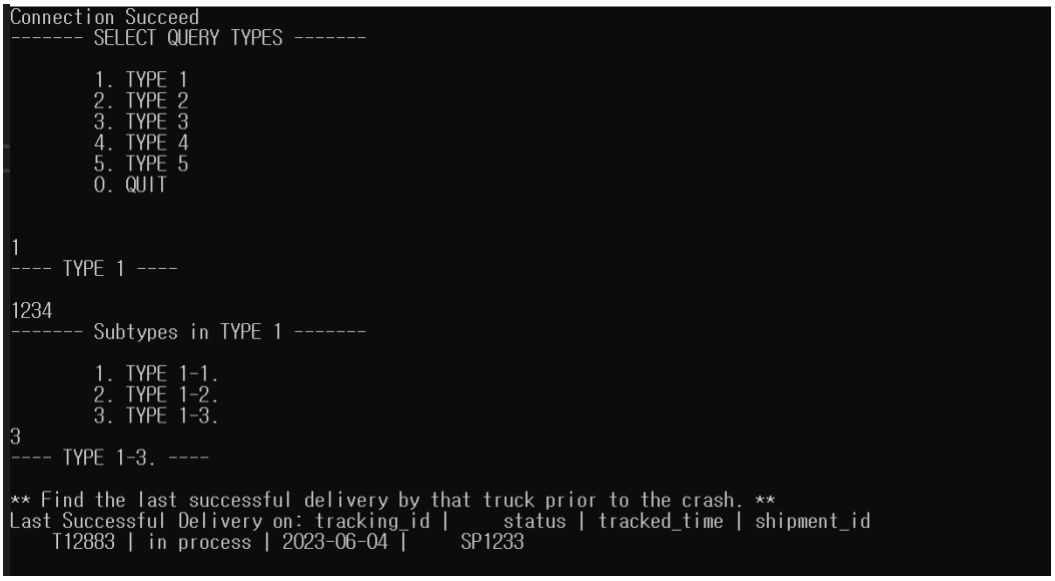
- subtype 3

Crush 된 트럭에서 마지막으로 배달된 항목을 찾는 것이므로 tracked_time 이 최댓값인 걸 찾으면 된다. Crash 된 트럭은 더 이상 배달을 수행하지 않기 때문이다. 따라서 사용된 query 는 다음과 같다.

```
string query1_3 = "select * from tracking where tracked_time = (select
MAX(tracked_time) from tracking) and vehicle_type = 'truck' and serial_number =
'";

query1_3 = query1_3 + truck + "'";
```

수행한 결과는 다음과 같다.



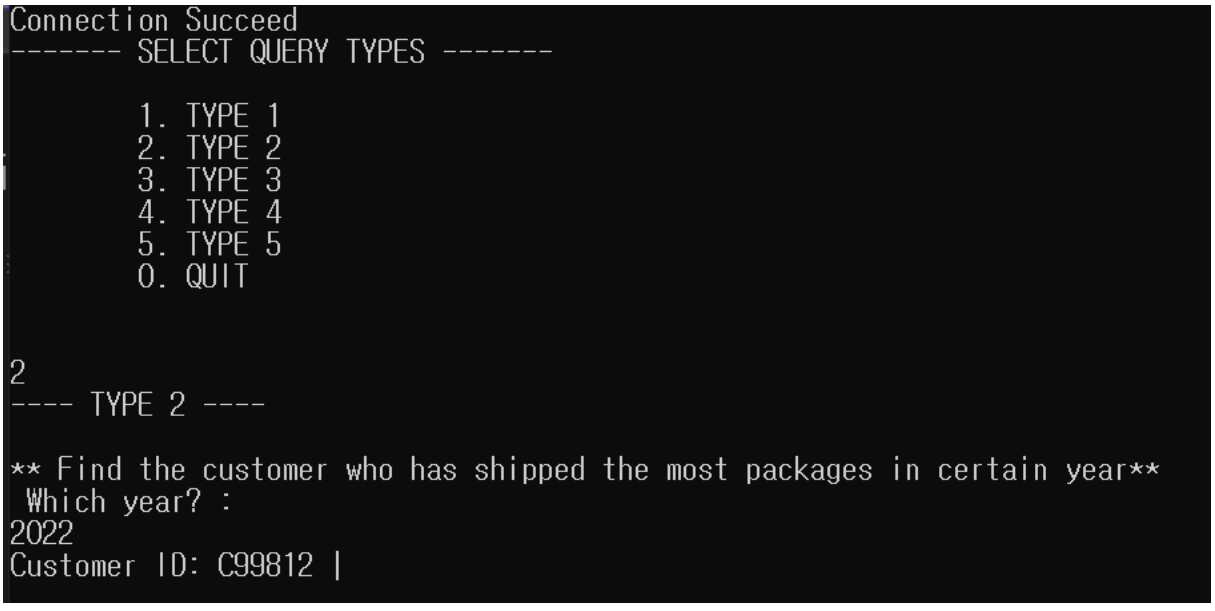
(2) TYPE 2

작년의 정보를 불러오는 것이므로 올해(2023)년 기준 2022 년의 정보를 불러와야 한다.

Type2 에서 사용된 query 문은 다음과 같다.

```
string query2 = "select customer_id from customer where last_year_frequency =
(select MAX(last_year_frequency) from customer)";
```

다음과 같은 결과를 반환한다.



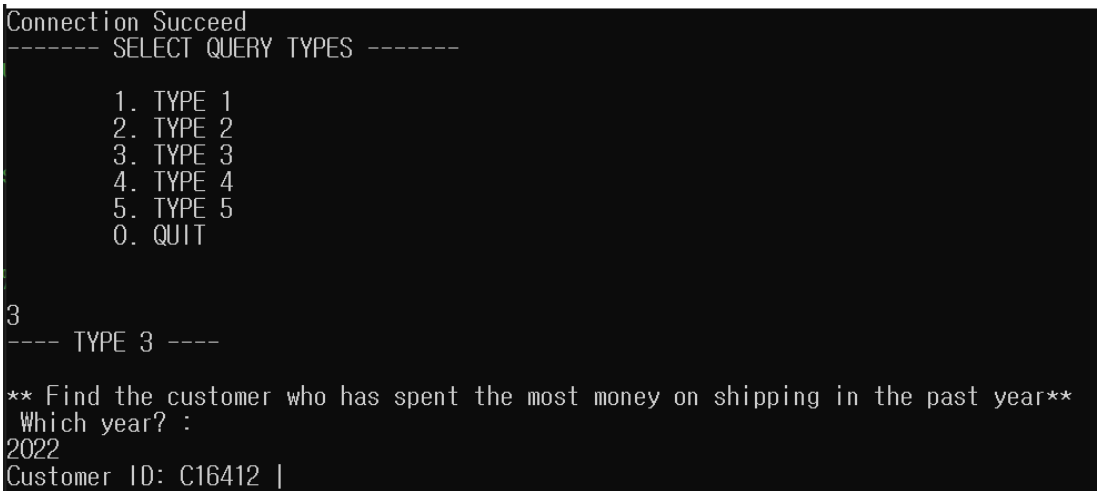
(3) TYPE 3

작년의 정보를 불러오는 것이므로 올해(2023)년 기준 2022 년의 정보를 불러와야 한다.

Type3 에서 사용된 query 문은 다음과 같다.

```
string query3 = "select customer_id from customer where last_year_total_charge
= (select MAX(last_year_total_charge) from customer)";
```

다음과 같은 결과를 반환한다.



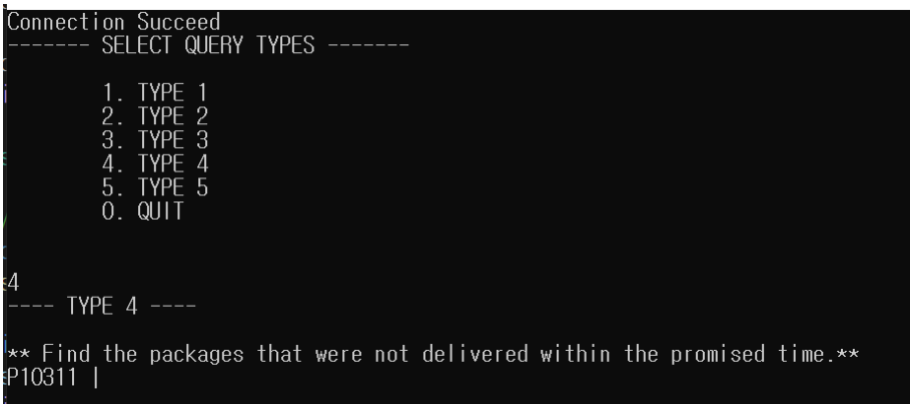
(4) TYPE 4

Package 에서 promised_time 과 delivered_time 을 비교해서 delivered time 이 늦은 경우를 반환해야 한다.

Type4 에서 사용된 query 문은 다음과 같다.

```
string query4 = "select package_id from package P, shipment S where
P.shipment_id = S.shipment_id and S.delivered_time>S.promised_time";
```

다음과 같은 결과를 반환한다.



(5) TYPE 5

입력된 날짜와 연도를 비교하여 bill 의 모든 정보를 불러오고, bill 의 타입별로 다른 정보를 출력한다.

Type4 에서 사용된 query 문은 다음과 같다.

```
string query5 = "select * from Bill where (select month(issued_date) = '";
query5 += month;
query5 += "') and (select year(issued_date) = '";
query5 += year;
query5 += "')";
```

다음과 같은 결과를 반환한다.

```
Connection Succeed
------- SELECT QUERY TYPES -------

        1. TYPE 1
        2. TYPE 2
        3. TYPE 3
        4. TYPE 4
        5. TYPE 5
        0. QUIT


5
---- TYPE 5 ----

** Generate the bill for each customer for the past month. Consider creating several types of bills.**
 Which year? :
2023
 Which month? :
02

 -- query: select * from Bill where (select month(issued_date) = '02') and (select year(issued_date) = '2023')


-------- A Simple Bill ------------

        Bill_ID |     Customer_ID |          Amount
         BC0988 |          C11012 |            9000
         BS1181 |          C11209 |          110000



-------- A Bill listing charges by type of service ------------

        Bill_ID |            Type |          Amount
         BC0988 |          Simple |            9000
         BS1181 |          Simple |          110000


-------- An Itemize billing ------------

        Bill_ID |     Customer_id |      Package_id |          Method |          Charge
         BC0988 |          C11012 |          P11111 |     Credit Card |            9000
         BS1181 |          C11209 |          P11311 |            Cash |          110000
```

(6) Quit

각각의 단계에서 0 을 누르면 해당 단계를 빠져나와 이전 단계로 돌아가도록 설계하였다. 또한 가장 첫번째 단계에서 0 을 누르면 프로그램을 종료하는 것과 동시에 입력되었던 data 와 table 을 모두 삭제할 수 있도록 두번째 Txt 파일을 불러와 query 를 수행하도록 하였다. 따라서 모든 단계가 종료되면 다음과 같이 table 이 삭제된 걸 확인할 수 있다.