데이터베이스시스템

CSE4110-02

〈Project #2〉

Normalization and Query Processing

서강대학교 경제학과

20160563 송진아

목 차

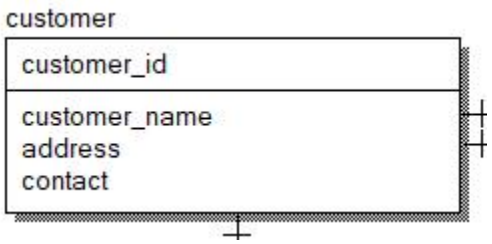
1 프로젝트 개요

이 프로젝트는 지난 첫 번째 프로젝트에서 작성한 Package Delivery System 에 관한 관계형 데이터베이스를 정규화하고 실제 데이터베이스를 구축한 뒤 목적에 맞는 Query 를 작성하고 잘 동작하는지 확인하는 것을 목표로 한다. 구체적으로 서술하면 다음과 같다. 정규화는 BCNF 를 만족하는 것을 기준으로 한다. Logical Shema 를 기반으로 BCNF Decomposition 을 마친 뒤, 이를 바탕으로 ERwin 의 Physical Shema 를 디자인한다. Physical Shema 는 Logical Shema 에 Data Types, Domain, Constraints, Relationship Type 을 추가하는 방식으로 작성한다. 여기까지 마치면, Physical Shema 를 바탕으로 실제 데이터베이스의 Table 을 Create 한다. 그 다음 Table 에 적합한 Data 를 Insert 한다. 마지막으로 주어진 조건을 충족하는 데이터를 가져오기 위한 Query 를 작성한다.

2 BCNF Decomposition

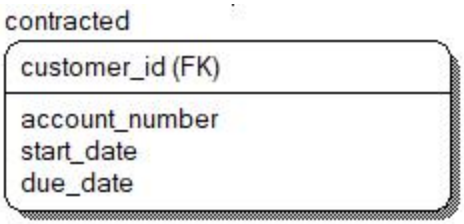
첫 번째 프로젝트에서 작성한 Logical Shema 의 relation 들이 BCNF 를 만족하는지 살펴보고 위배된다면 Decomposition 한다. 교재『Database System Concepts 7th edition, McGraw-Hill Book Company』7.5.1 장의 BCNF Testing 과 Decomposition 방식을 따른다.

- 'customer' relation



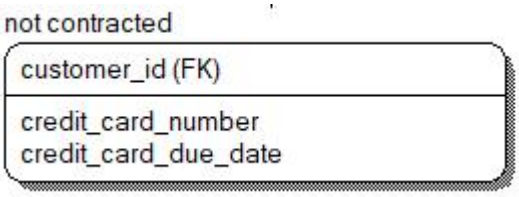
Primary Key 로 지정된 'customer_id'는 각 고객마다 고유하게 부여된 값이므로 'customer_name', 'address', 'contact'를 결정할 수 있다. 반면 'customer_name'은 동명이인이 존재할 가능성이 있으므로 'customer_id', 'address', 'contact'를 결정할 수 없다. 'address' 또한 동일한 배송지에 여러 명이 함께 살아서 각자의 택배를 주문하는 경우나, 이사를 가서 주소지가 변경되었지만 업데이트하지 않은 경우로 중복된 주소지가 생기는 경우가 생길 수 있다. 전화번호의 경우 사무실 번호를 공유하거나 번호를 변경하였지만 업데이트 하지 않은 경우에 서로 다른 고객 간에 동일한 연락처를 사용할 수 있다. 따라서 'customer_id'를 제외한 나머지 attributes 는 다른 attributes 를 결정하지 않는다. 즉 BCNF 를 충족한다.

- 'contracted' relation



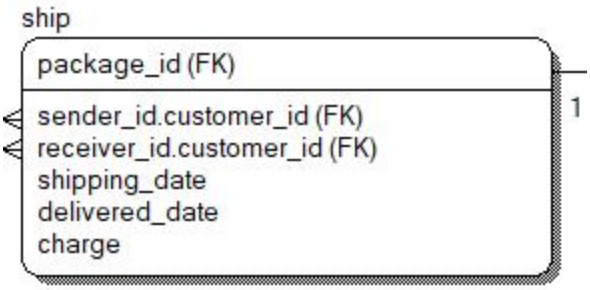
'customer_id'는 앞서 서술했다시피 각 고객의 고유번호이므로 Primary Key 가 된다. 그러나 'account_number'는 회사 계좌번호를 이용하거나 가족 명의의 통장을 공유하는 경우 때문에 서로 다른 고객 간에 중복이 생길 수 있다. 'start_date'와 'due_date'도 동일한 날짜에 계약을 맺거나, 동일한 날짜에 계약을 종료하는 경우가 빈번하므로 중복이 생긴다. 따라서 'customer_id'를 제외한 나머지 attributes 는 다른 attributes 를 결정하지 않는다. 즉 BCNF 를 충족한다.

- 'not contracted' relation



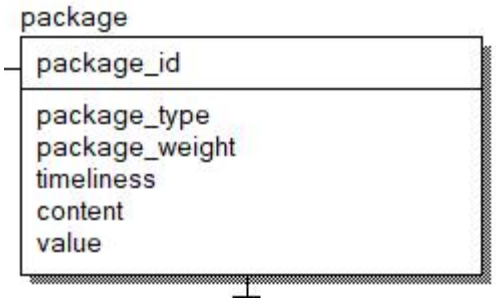
'customer_id'는 primary key가 되지만, 'credit_card_number'는 계좌번호와 마찬가지 이유로 다른 attributes를 결정하지 못하고, 'credit_card_due_date'는 'due_date'와 마찬가지 이유로 다른 attributes를 결정하지 못한다. trivial하지 않다면 primary key(super key)에서 나머지 attibutes로 가는 functional dependency만 존재하므로 BCNF를 충족한다.

- 'ship' relation



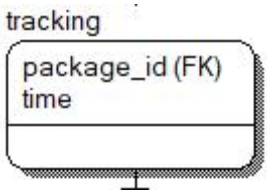
'package_id'는 'customer_id'와 유사하게 각 package 마다 고유하게 부여되는 값이다. 따라서 'package_id'만 있으면 각 배송마다 발신자와 수신자, 배송 시작일, 도착일, 배송비를 특정할 수 있다. 따라서 'package_id'에서 다른 attributes로 가는 functional dependency가 존재하고 이는 primary key(super key)이므로 BCNF를 만족한다. 나머지 attributes들에 대해서는 한 고객이 여러 번 발송을 할 수도 있고 한 고객이 여러 번 수신을 할 수도 있으며, 동일한 날짜에 여러 건의 택배가 출발하거나 도착하고 동일한 요금의 택배가 존재하므로 super key가 아니면서 다른 attributes로의 functional dependency가 없다. 따라서 BCNF를 충족한다.

- 'package' relation



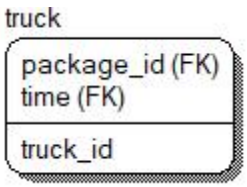
'package_id'는 고유한 값이므로 package 를 특정할 수 있다. 반면 나머지 attributes 는 조합하여도 package 를 결정하지 못한다. 'package_id'만 다르고 나머지 attributes 는 전부 같은 다른 package 가 존재할 수 있기 때문이다. 따라서 non-trivial 하다면 'package_id'에서 나머지 attributes 로의 functional dependency 만이 존재하며 'package_id'는 super key 이므로 BCNF 를 충족한다.

- 'tracking' relation



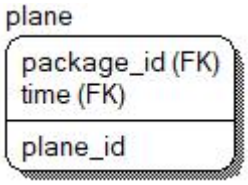
tracking 의 경우, 'package_id'와 'time'이 모두 있어야 특정된다. 하나의 package 당 여러 개의 추적 기록이 존재하며, 같은 시간에 여러 개의 package 가 기록되기 때문에 하나의 attribute 만으로는 결정할 수 없다. 하나의 package 가 동일 시점에 여러 곳에서 존재할 수는 없으므로 'package_id'와 'time' 두개를 합하면 추적 기록을 특정할 수 있다. 이 경우 모든 attributes 가 primary key 이므로 BCNF 가 충족된다.

- 'truck' relation



truck 의 경우, tracking 의 specialization 이므로 'package_id'와 'time'에 의해 특정된다. 하나의 package 가 하나의 트럭만 거친다는 보장이 없고 동일한 시점에 하나의 트럭만 운행되지는 않으며 트럭 번호만 가지고 소포나 시간을 특정하지 못하므로 non-tirivial 한 것에 한정하면 'package_id'와 'time'에서 'truck_id'로 가는 functional dependency 만 존재하고 이는 BCNF 를 만족한다.

- ‘plane’ relation



‘truck’ relation 과 동일한 이유로 BCNF 를 만족한다.

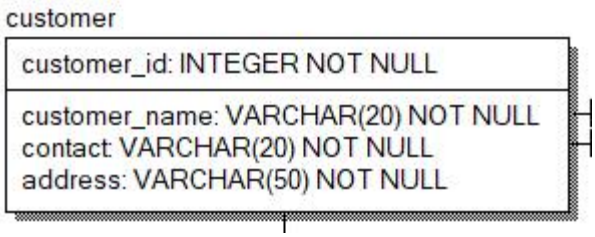
- ‘warehouse’ relation

‘truck’ relation 과 동일한 이유로 BCNF 를 만족한다.

3 Physical Schema Diagram

‘2 BCNF Decomposition’에서 살펴본 Logical Shema 가 BCNF 를 만족하는 것을 확인하였으므로, 기존의 Logical Shema 에 Data Types, Domain, Constraints, Relationship Type 을 추가하여 Physical Shema Diagram 을 작성한다.

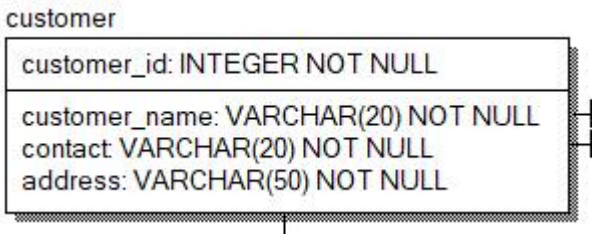
- ‘customer’ relation



‘customer_id’는 숫자로 구성하기 위해 integer 로 지정하였다. Primary Key 이므로 Not Null 로 지정하였다. 고객 이름, 연락처(전화번호), 주소지도 택배를 보내거나 받을 때 필수 정보이므로 Not Null 로 지정하였다. 이름의 경우 Character String, 전화번호의

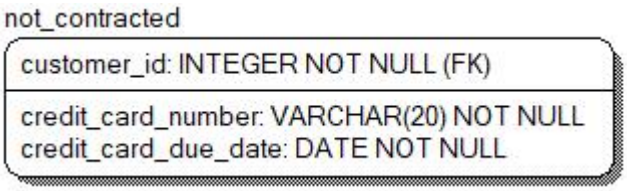
경우 국가 코드 표기(예를 들어 +82)를 위해 Character String 으로 지정하였으며 주소도 Character String 이다.

- ‘contract’ relation



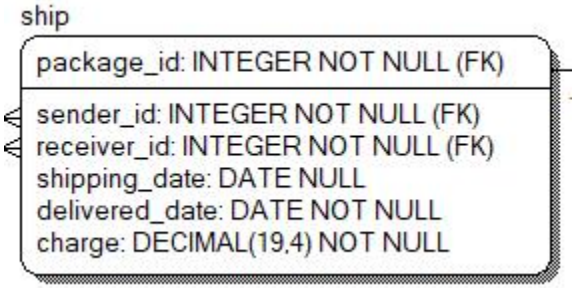
계약 시작일과 계약 종료일을 의미하는 ‘start_date’와 ‘due_date’의 경우 구체적인 시간 정보까지는 불필요하고 날짜 정보까지를 저장하기 위해 Date 타입으로 지정하였으며 ‘account_number’는 ‘-’를 포함할 수 있으므로 Character String 으로 지정하였다. 이들 정보는 모두 계약에 필수적이라고 판단하여 Not Null 로 지정하였다.

- ‘not contracted’ relation



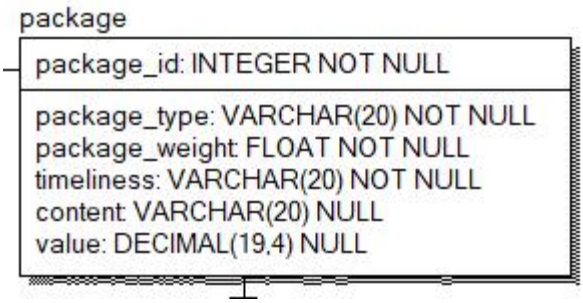
신용카드 번호는 ‘-’를 포함할 수 있으므로 Character String 으로 지정하였으며 신용카드 유효기간은 Date 로 지정하였다. 실제로는 만료 연월만 기입하지만 일자는 해당월 말일로 저장하기 위함이다. 이들은 택배비 결제시에 신용카드 유효성을 체크하기 위해서 모두 필요한 정보이므로 Not Null 로 지정하였다.

- ‘ship’ relation



'package_id'는 숫자로 저장하기 위해 Integer 로 설정하였으며 Primary Key 이므로 Not Null 이다. 수신인, 발신인은 'customer_id'의 foreign key 이므로 동일하게 Integer 이다. 'shipping_date'와 'delievered_date'는 날짜를 저장하기 위해 Date 이고, 'charge'의 경우 금액이 소수점까지 표기(예를 들어 $9.99)될 수 있으므로 Decimal(19.4)로 지정하였다. 이들 정보는 'delivered_date'를 빼고 필수적이라고 판단하였는데, 도착 정보는 택배가 유실되는 경우 존재하지 않을 수도 있기 때문에 Null 을 허용하였다.

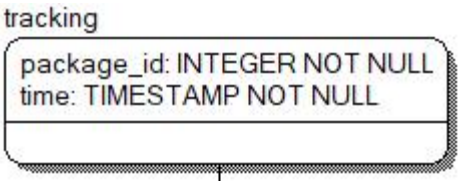
- 'tracking' relation



'package_type'은 flat envelope, small box, larger boxes 를 저장하기 위해 Character String 으로 지정하였고, 'package_weight'는 소수점을 표기하기 위해(예를 들어 1.5kg) Float 로 지정하였고, 'timeliness'는 overnight, second day, or longer 를 저장하기 위해 Character String 으로 지정하였다. 또 'content'는 고객이 어떠한 물건인지 기입하므로 Character String, value 는 상품 가액을 지정하기 위해 'charge'와 같이 Decimal(19.4)로 지정하였다. 'package_type', 'package_weight', 'timeliness'의 경우 배송비를 책정하기 위해 필요한 정보이므로 Not Null 로 설정하였으며 'content'와

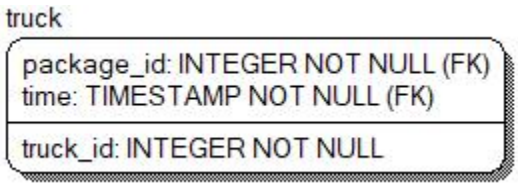
‘value’의 경우 취급 주의 상품인 경우와 같이 특수한 경우에만 필요하므로 Null 을 허용하였다.

- ‘tracking’ relation



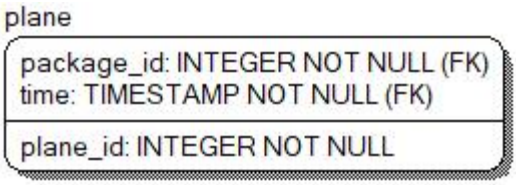
‘time’의 경우 세밀한 추적을 위해 Date 가 아닌 시 분 초까지 기록되는 Timestamp 로 지정하였다. 이들은 Primary Key 이므로 Not Null 이다.

- ‘truck’ relation



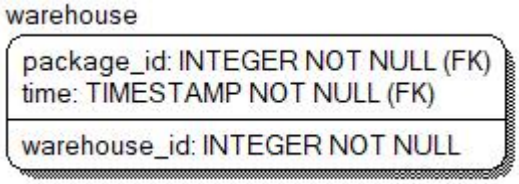
‘truck_id’의 경우 차량 번호를 기록하기 위해 Integer 로 지정하였으며 위치가 기록되지 않으면 추적이 의미 없다고 판단하여 Not Null 로 설정하였다.

- ‘plane’ relation



‘plane_id’의 경우 기체 번호를 기록하기 위해 Integer 로 지정하였으며 위치가 기록되지 않으면 추적이 의미 없다고 판단하여 Not Null 로 설정하였다.

- ‘warehouse’ relation



'warehouse _id'의 경우 차량 번호를 기록하기 위해 Integer 로 지정하였으며 위치가 기록되지 않으면 추적이 의미 없다고 판단하여 Not Null 로 설정하였다.

- constraints, relationship type

'5 부록'에 첨부한 Physical Shema Diagram 을 참조한다. constraints 와 relationship type 을 이와 같이 설정한 이유는 첫 번째 프로젝트의 보고서에 상세히 기술하였으므로 생략한다.

4 Queries

한 Relation 마다 최소 15 개의 Tuples 를 Insert 하였으며, 주어진 Type I 부터 Type V 를 처리하기 위한 Query 를 작성하여 해당되는 Data 를 추출하였다. x64 환경에서 데이터베이스를 처리하였다.

- Type I

1721 트럭이 2021 년 5 월 1 일 7 시에 출발하여 이동 중 당일 10 시 20 분에 사고가 났다고 가정하였다.

Type I-1 의 경우 다음과 같이 Query 문을 작성하였다.

"select customer_name
from customer, ship, truck
where truck_id = 1721 and ship.package_id = truck.package_id and _time 〉

'2021-05-01 07:00:00' and _time < '2021-05-01 10:20:00' and sender_id = customer_id;"

사고 시점 사이에 1721 트럭에 기록이 있는 택배를 찾고 발송인을 찾는다.

Type I-2 의 경우 다음과 같이 Query 문을 작성하였다.

"select customer_name
from customer, ship, truck
where truck_id = 1721 and ship.package_id = truck.package_id and _time > '2021-05-01 07:00:00' and _time < '2021-05-01 10:20:00' and receiver_id = customer_id;"

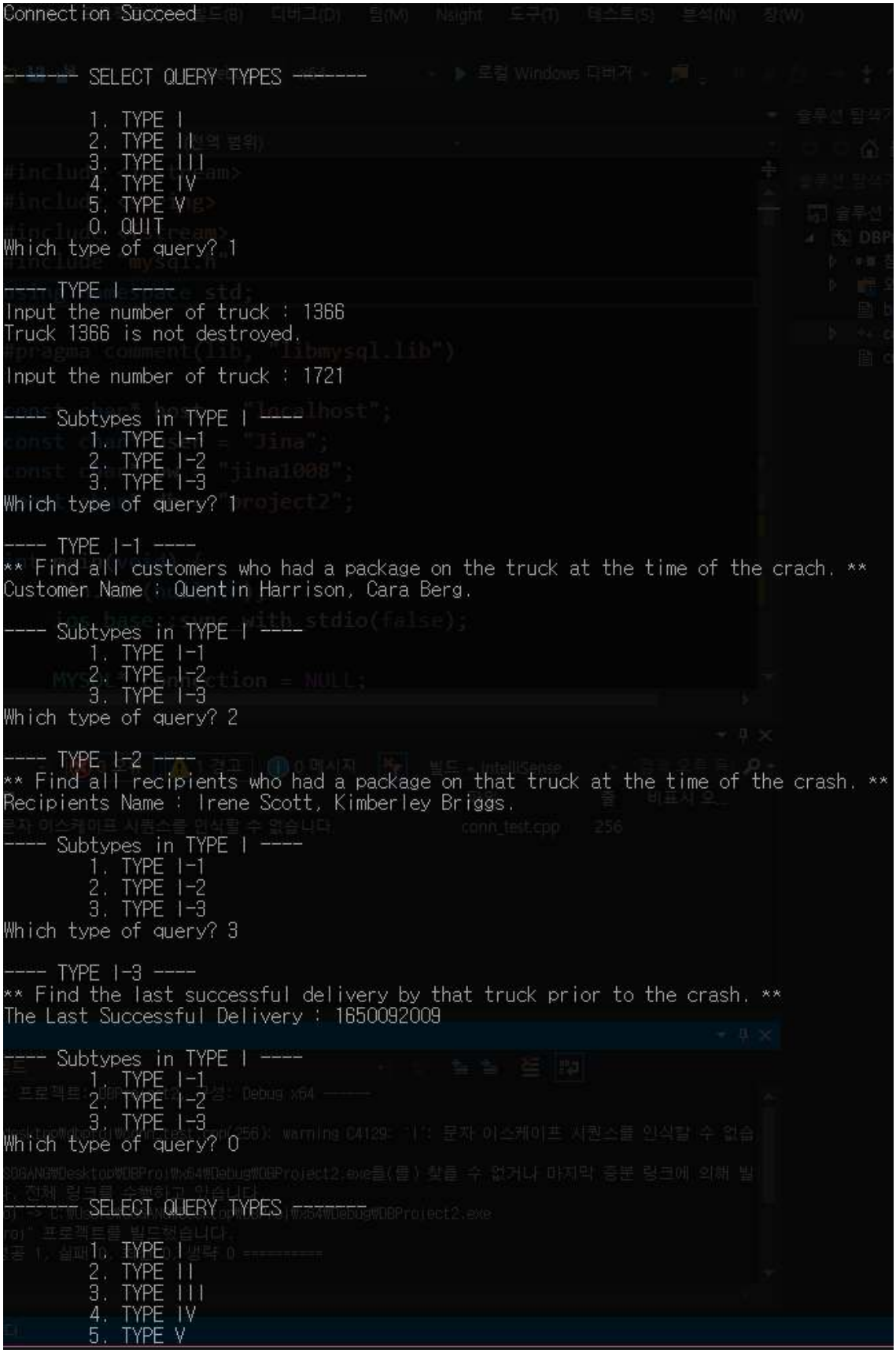
사고 시점 사이에 1721 트럭에 기록이 있는 택배를 찾고 수신인을 찾는다.

Type I-3 의 경우 다음과 같이 Query 문을 작성하였다.

"select S.package_id
from ship as S, truck as T
where S.delivered_date=(select max(delivered_date)
    from ship, truck
    where truck_id = 1721 and ship.package_id = truck.package_id and delivered_date < '2021-05-01 10:20:00')
and T.truck_id = 1721 and S.package_id = T.package_id;"

사고 시점 이전에 1721 트럭에 기록이 있는 택배 중 가장 최근 기록의 시점을 찾고 그 시점에 1721 트럭에 있었던 택배의 번호를 찾는다.

다음과 같이 모두 올바른 데이터를 추출해낸 것을 확인할 수 있었다.


Connection Succeed
------- SELECT QUERY TYPES -------
1. TYPE I
2. TYPE II
3. TYPE III
4. TYPE IV
5. TYPE V
0. QUIT
Which type of query? 1
---- TYPE I ----
Input the number of truck : 1366
Truck 1366 is not destroyed.
Input the number of truck : 1721
---- Subtypes in TYPE I ----
1. TYPE I-1
2. TYPE I-2
3. TYPE I-3
Which type of query? 1
---- TYPE I-1 ----
** Find all customers who had a package on the truck at the time of the crach. **
Customer Name : Quentin Harrison, Cara Berg.
---- Subtypes in TYPE I ----
1. TYPE I-1
2. TYPE I-2
3. TYPE I-3
Which type of query? 2
---- TYPE I-2 ----
** Find all recipients who had a package on that truck at the time of the crash. **
Recipients Name : Irene Scott, Kimberley Briggs.
---- Subtypes in TYPE I ----
1. TYPE I-1
2. TYPE I-2
3. TYPE I-3
Which type of query? 3
---- TYPE I-3 ----
** Find the last successful delivery by that truck prior to the crash. **
The Last Successful Delivery : 1650092009
---- Subtypes in TYPE I ----
1. TYPE I-1
2. TYPE I-2
3. TYPE I-3
Which type of query? 0
------- SELECT QUERY TYPES -------
1. TYPE I
2. TYPE II
3. TYPE III
4. TYPE IV
5. TYPE V

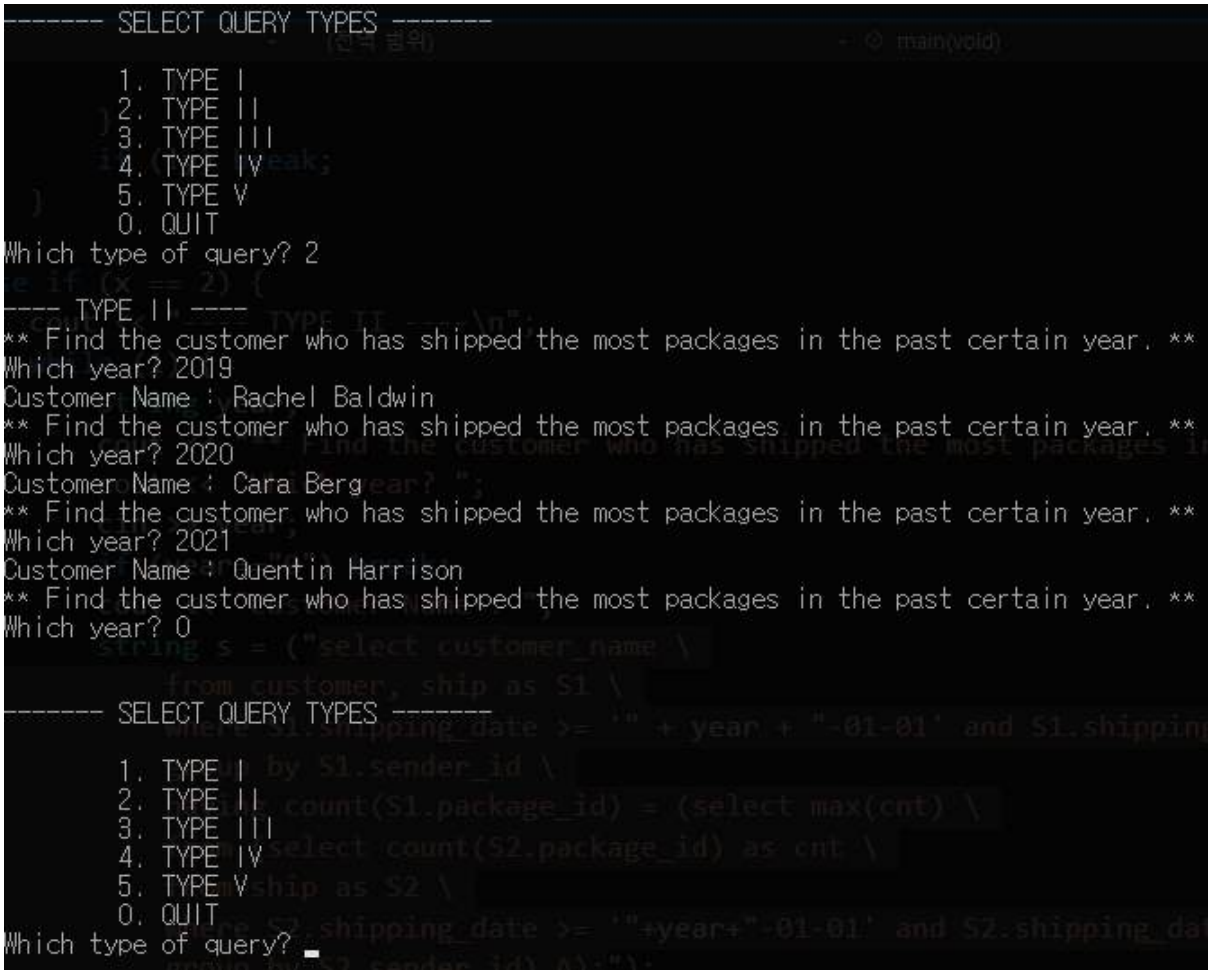
- Type II

다음과 같이 Query 문을 작성하였다.

```
“select customer_name
from customer, ship as S1
where S1.shipping_date 〉= '" + year + "-01-01' and S1.shipping_date 〈= '" +
year + "-12-31' and S1.sender_id = customer_id
group by S1.sender_id
having count(S1.package_id) = (select max(cnt)
    from (select count(S2.package_id) as cnt
        from ship as S2
        where S2.shipping_date 〉= '"+year+"-01-01' and S2.shipping_date 〈=
        '"+year+"-12-31'
        group by S2.sender_id) A);”
```

년도를 입력 받고 해당 년도 1 월 1 일부터 12 월 31 일 사이의 배송 건들을 보낸 사람을 기준으로 그룹화한다. 사람 별로 발송 건수를 세고 그 중 최대값을 뽑는다. 최대값에 해당하는 횟수만큼 발송한 사람의 이름을 찾는다. 만약 발송 횟수가 가장 많은 사람이 여러명이라면 모두 구한다.

다음과 같이 결과가 잘 나오는 것을 확인할 수 있었다.



- Type III

다음과 같이 Query 문을 작성하였다.

"select customer_name

from customer, ship as S1

where S1.shipping_date >= '" + year + "-01-01' and S1.shipping_date <= '" +

year + "-12-31' and S1.sender_id = customer_id

group by S1.sender_id

having sum(S1.charge) = (select max(spent)

from (select sum(S2.charge) as spent

from ship as S2

where S2.shipping_date >= '" + year + "-01-01' and

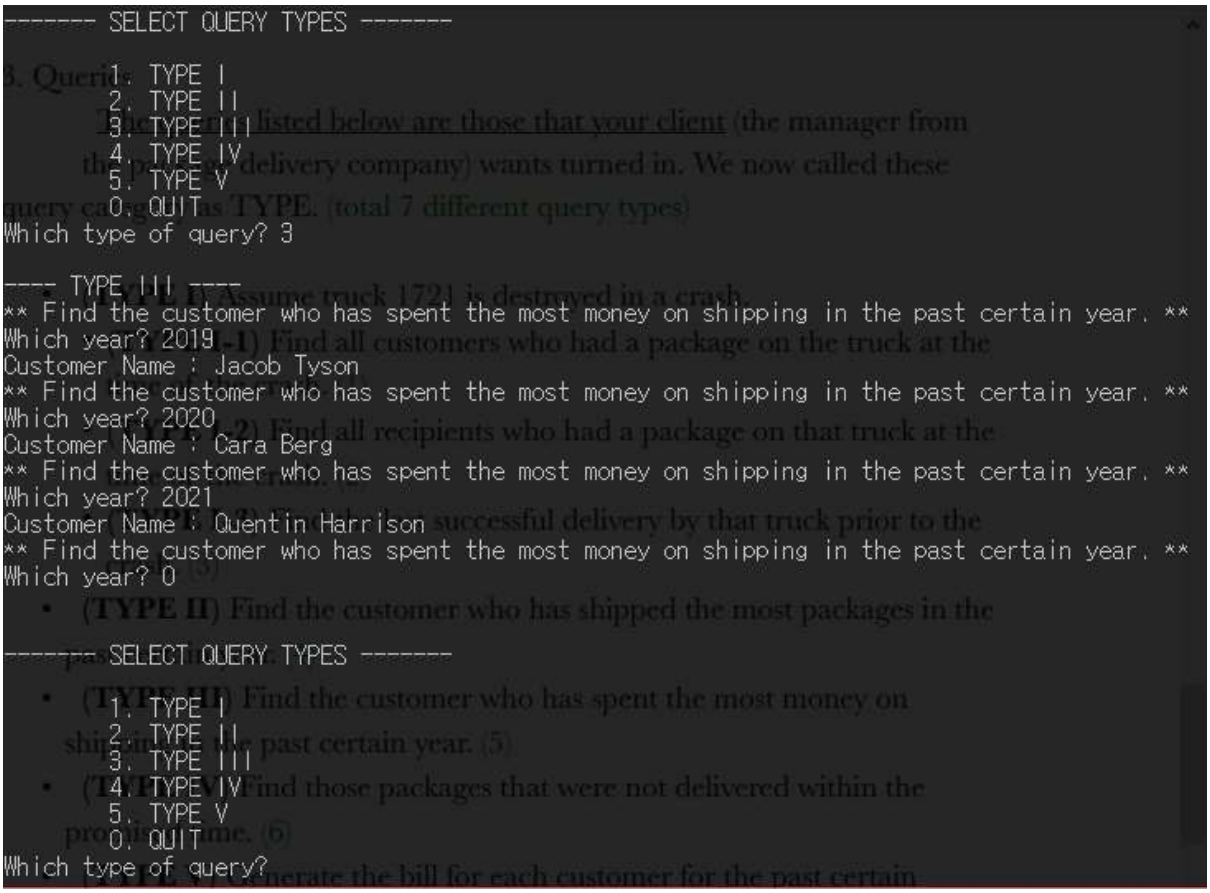
S2.shipping_date

〈= '" + year + "-12-31'

group by S2.sender_id) A);"

년도를 입력 받고 해당 년도 1 월 1 일부터 12 월 31 일 사이의 배송 건들을 보낸 사람을 기준으로 그룹화한다. 사람 별로 배송 건당 비용의 합을 구하고 그 중 최대값을 뽑는다. 최대값에 해당하는 금액만큼의 비용을 지불한 사람의 이름을 찾는다. 만약 가장 많은 비용을 지불한 사람이 여러명이라면 모두 구한다.

다음과 같이 결과가 잘 나오는 것을 확인할 수 있었다.



- Type IV

다음과 같이 Query 문을 작성하였다.

"select ship.package_id

from ship, package

where ship.package_id = package.package_id and

((timeliness = 'overnight' and shipping_date < delivered_date

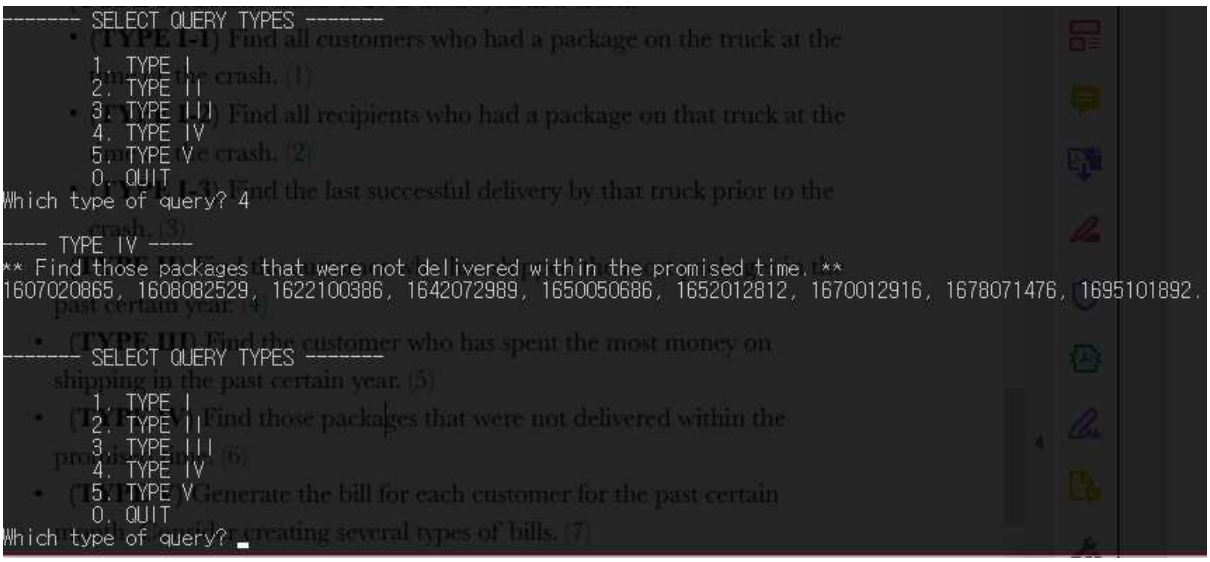
or (timeliness = 'second day' and date_add(shipping_date, interval 1 day) <

delivered_date));"

timeliness 의 종류는 overnight, second day, or longer 로 세 가지이다.

overnight 이면서 당일까지 배송되지 않았거나, second day 이면서 다음 날까지 배송되지 않은 경우의 'package_id'를 찾았다. 이외의 경우는 배송일을 지정하지 않은 것으로 간주하여 제외하였다.

다음과 같이 결과가 잘 나온 것을 확인하였다.



- Type V

다음과 같이 Query 문을 작성하였다.

우선 Simple Bill 을 위한 Query 이다.

```
"select address, sum(charge)
from customer, ship
where customer_name = '" + name + "'
and customer_id = sender_id
and shipping_date >= '" + month + "-01'
and shipping_date <= '" + month + "-31';"
```

입력 받은 이름과 연월에 해당하는 배송 건들을 찾아 배송비의 총액과 해당인의 주소를 구하였다.

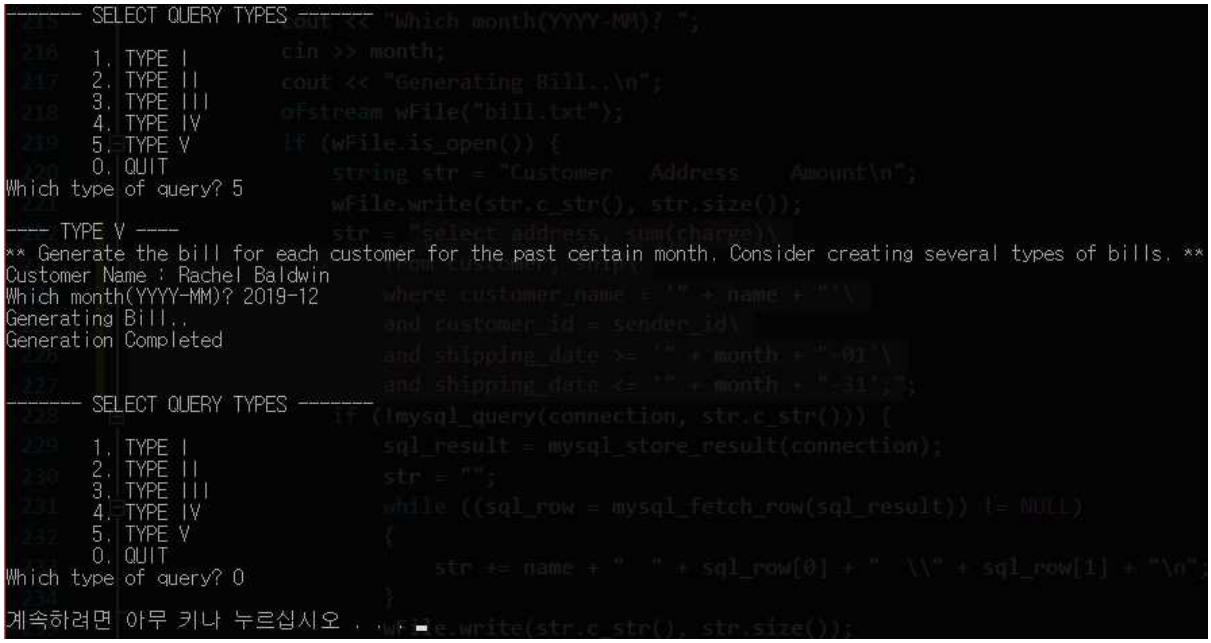
다음으로 itemize billing 과 각 서비스에 대한 비용을 구하기 위한 Query 이다.

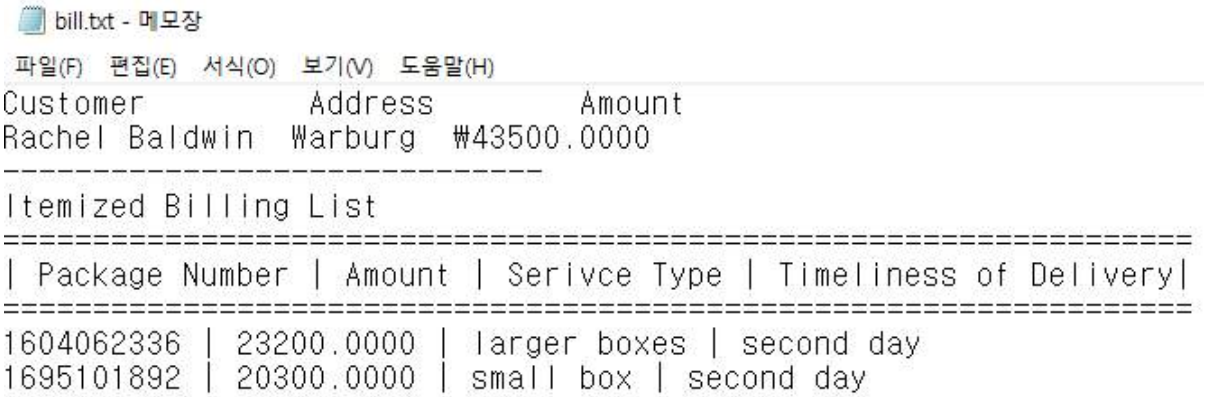
```
"select package.package_id, charge, package_type, timeliness
from customer, package, ship
where customer_name = '" + name + "'
and customer_id = sender_id
and shipping_date >= '" + month + "-01'
and shipping_date <= '" + month + "-31'
and package.package_id = ship.package_id;"
```

입력 받은 이름과 연월에 해당하는 배송 건들을 찾아 ‘package_id’, ‘charge’, ‘package_type’, ‘timeliness’를 구하였다.

이 두 Queries 에서 얻은 정보를 적절한 형식으로 파일에 썼다.

결과는 다음과 같이 잘 나왔다.


------- SELECT QUERY TYPES -------
1. TYPE I
2. TYPE II
3. TYPE III
4. TYPE IV
5. TYPE V
0. QUIT
Which type of query? 5
---- TYPE V ----
** Generate the bill for each customer for the past certain month. Consider creating several types of bills. **
Customer Name : Rachel Baldwin
Which month(YYYY-MM)? 2019-12
Generating Bill..
Generation Completed
------- SELECT QUERY TYPES -------
1. TYPE I
2. TYPE II
3. TYPE III
4. TYPE IV
5. TYPE V
0. QUIT
Which type of query? 0
계속하려면 아무 키나 누르십시오 . . .


bill.txt - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
Customer Address Amount
Rachel Baldwin Warburg ₩43500.0000
Itemized Billing List
| Package Number | Amount | Serivce Type | Timeliness of Delivery|
1604062336 | 23200.0000 | larger boxes | second day
1695101892 | 20300.0000 | small box | second day

5 부록

- schema diagram (physical)