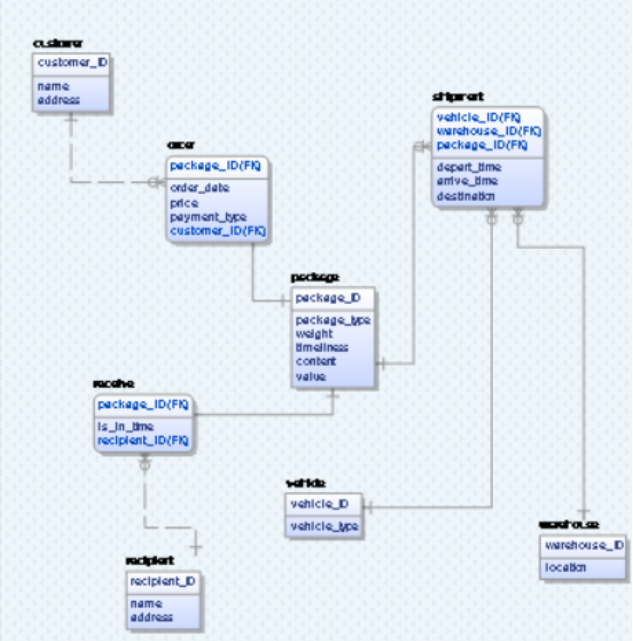
Project 2 보고서

20190963 한다현

1. BCNF Decomposition

먼저 프로젝트 1에서 설계한 logical schema diagram에서 shipment table을 수정하였는데, 수정된 logical schema diagram은 다음과 같다.



기존에 depart_time, depart_date였던 속성들을 depart_time으로 통합하였고, arrive_time, arrive_date였던 속성들을 arrive_date로 통합하였다.

위의 Logical schema diagram에서 BCNF가 아닌 table은 shipment이다. 이를 제외한 다른 테이블은 Functional Dependencies의 알파가 superkey이기 때문에 BCNF를 만족한다.

Shipment table에서 F는

{vehicle_ID, warehouse_ID, depart_time -> arrive_time, destination,

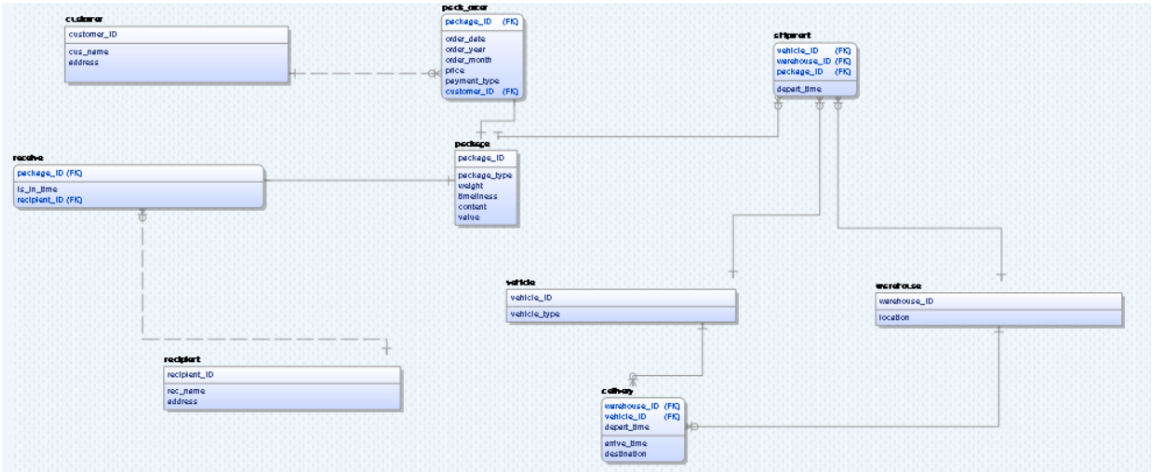
vehicle_ID, warehouse_ID, package_ID -> depart_time, arrive_time, destination}이다.

vehicle_ID, warehouse_ID, depart_time -> arrive_time, destination를 R1,

vehicle_ID, warehouse_ID, package_ID는 superkey이지만, vehicle_ID, warehouse_ID, depart_time는 superkey가 아니다. 따라서 shipment를 두 개의 relation schema R1, R2로 decompose하면

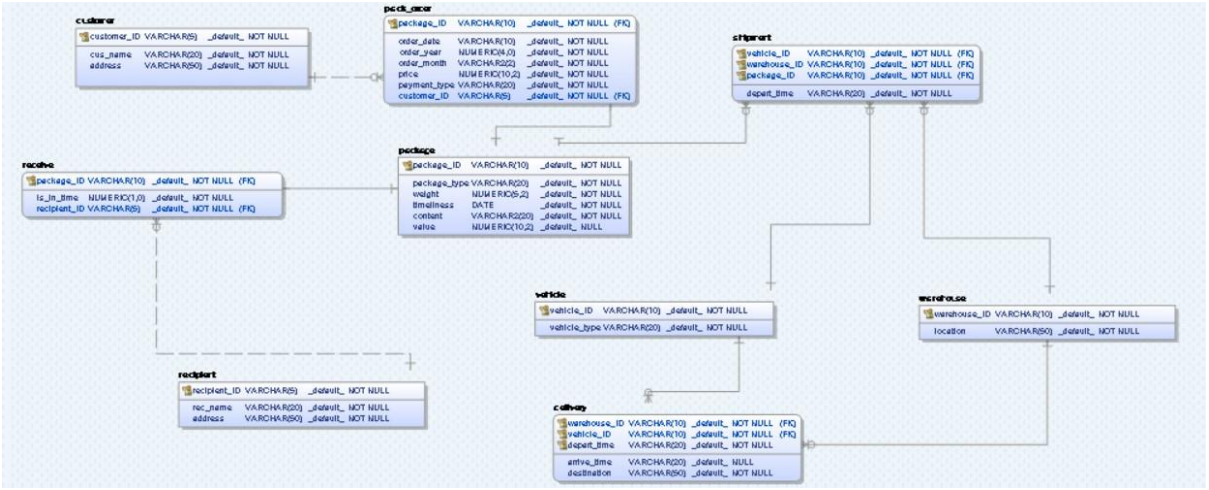
R1 = (vehicle_ID, warehouse_ID, depart_time, arrive_time, destination)

R2 = (vehicle_ID, warehouse_ID, package_ID, depart_time) 이다. R1을 delivery로, R2를 shipment인 Logical schema diagram은 다음과 같다.



2. Physical schema diagram

프로젝트 2의 Physical schema diagram은 다음과 같다.



2-1. Customer

Customer는 고객에 대한 정보를 저장하는 relation schema로, 세 개의 속성을 갖는다. customer_ID는 customer table의 primary key이며, 고객의 ID를 의미하고 최대 5 길이의 문자형이다. cus_name은 고객의 이름을 의미하고, null 값을 허용하지 않는다. cus_name의 데이터 타입은 최대 길이 20의 문자형이며 SQL query를 작성할 때 name이라는 커맨드가 존재하기 때문에 name이 아닌 cus_name으로 수정하였다. address는 고객의 주소를 의미하며 최대 길이 50의 문자형이고, null 값을 허용하지 않는다.

2-2. recipient

recipient는 수신자에 대한 정보를 저장하는 relation schema로, 세 개의 속성을 갖는다. recipient_ID는 수신자의 ID를 의미하고, 최대 길이가 5인 문자형이며 recipient table의 primary key이다. rec_name은 수신자의 이름을 의미하고, null 값을 허용하지 않는다. rec_name의 데이터 타입은 최대 길이 20의 문자형이며 SQL query를 작성할 때 name이라는 커맨드가 존재하기 때문에 name이 아닌 rec_name으로 수정하였다. address는 수시자의 주소를 의미하며 최대 길이 50의 문자형이고, null 값을 허용하지 않는다.

2-3. package

package는 패키지에 대한 정보를 저장하는 relation schema로, 6개의 속성을 갖는다. package_ID는 패키지의 ID를 의미하고, package table의 primary key이며 최대 길이 10인 문자형이다. package_type은 패키지의 타입, 즉 small box, large box, envelope인지를 구분하는 속성이며 최대 길이 20인 문자형으로 null 값을 허용하지 않는다. weight는 패키지의 무개이며 총 길이 5, 소수점 아래가 2자리인 숫자이고 null 값을 허용하지 않는다. timeliness는 패키지가 배송되어야 하는 날짜이다. DATE 형으로 저장하고 null 값을 허용하지 않는다. content는 패키지의 내용물이며 최대 길이 20인 문자형이고 null 값을 허용하지 않는다. value는 패키지 내용물의 가치이며 총 길이 10, 소수 2자리인 숫자이고, null 값을 허용한다.

2-4. pack_order

기존 Logical schema diagram에서는 order라는 이름의 테이블이었지만 SQL query에서 order라는 커맨드가 존재하기 때문에 pack_order로 수정하였다. pack_order는 패키지 주문에 대한 정보를 저장하는 relation schema로, 7개의 속성이 있다. package_ID가 primary key이고, package를 참조하는 foreign key이다. order_date는 최대 길이 10인 문자형이고 고객이 주문한 날짜를 저장한다. order_year는 order_date의 year 부분을 따로 저장하는 속성으로, 4자리 숫자이다. order_month는 order_date의 month 부분을 따로 저장하는 속성으로 2자리 숫자이다. price는 패키지 배송의 가격이며 총 길이 10, 소수 2자리인 숫자이다. payment_type은 고객이 결재하는 방식, 즉 credit card로 결재하였는지, monthly로 결재하였는지 등을 저장하는 속성으로 최대 길이 20인 문자형이다. customer_ID는 customer를 참조하는 foreign key이다. pack_order 테이블의 모든 속성은 null 값을 허용하지 않는다.

2-5. receive

receive는 패키지의 도착에 대한 정보를 저장하는 relation schema로, 3개의 속성을 갖는다. package_ID는 package를 참조하는 foreign key이자 primary key이다. is_in_time은 null 값을 허용하지 않는 1자리 숫자이고, 제시간에 도착하면 0, 그렇지 않으면 1을 저장하는데, 아직 배송이 완료되지 않은 패키지의 경우 0을 저장한다. recipient_ID는 recipient를 참조하는 foreign key이고,

null 값을 허용하지 않는다.

2-6. vehicle

vehicle은 운송 수단에 대한 정보를 저장하는 relation schema로, 2개의 속성을 갖는다. vehicle_ID는 해당 vehicle의 ID를 의미하며 최대 길이 10인 문자형이고 primary key이다. vehicle_type은 vehicle의 종류, 즉 트럭인지, 배인지, 비행기인지 등을 저장하며 최대 길이 20인 문자형이고, null 값을 허용하지 않는다.

2-7. warehouse

warehouse는 허브에 대한 정보를 저장하는 relation schema로, 2개의 속성을 갖는다. warehouse_ID는 warehouse의 primary key이고, 최대 길이 10인 문자형이다. location은 warehouse의 위치를 저장하는 속성으로, 최대 길이 50인 문자형이고 null 값을 허용하지 않는다.

2-8. shipment

shipment는 패키지의 이동에 대한 정보를 저장하는 relation schema로, 4개의 속성을 갖는다. vehicle_ID는 vehicle을 참조하는 FK이고, warehouse_ID는 warehouse를 참조하는 FK이고, package_ID는 package를 참조하는 FK이다. 이 세 개의 속성의 집합이 shipment의 PK이다. depart_time은 패키지가 운송 수단을 타고 허브를 출발하는 시간을 저장하는 속성이다. '2023-06-08 11:23:55'와 같은 형태로 저장되며 null 값을 허용하지 않는다.

2-9. delivery

delivery는 운송수단의 이동에 대한 정보를 저장하는 relation schema로, 5개의 속성을 갖는다. warehouse_ID는 warehouse를 참조하는 FK, vehicle_ID는 vehicle을 참조하는 FK이다. depart_time은 패키지가 운송 수단을 타고 허브를 출발하는 시간을 저장하는 속성이다. 이 세 속성의 집합이 delivery의 PK이다. arrive_time은 패키지가 목적지에 도착한 시간이다. 아직 패키지가 목적지에 도착하지 않은 경우 null 값을 허용한다. '2023-06-08 11:23:55'와 같은 형태로 저장되며 최대 길이 20인 문자형이다. destination은 목적지를 저장한다. 목적지는 다음 warehouse가 될 수도 있고, 수신자의 주소가 될 수도 있다. 최대 길이 50인 문자형이며 null 값을 허용하지 않는다.

3. ODBC Implementation

3-1. int main(void)

main 함수에서는 코드를 DB와 연결하고 성공적으로 연결된 경우, input text 파일을 읽고 DB에 저장하는 기능을 수행한다. Input text 파일은 20190963.txt 파일과 20190963_del.txt 파일 두 가지가 있는데, 20190963.txt 파일은 테이블을 생성하고 데이터를 추가하는 create 문과 insert 문이 포함되어 있고, 20190963_del.txt 파일을 테이블과 데이터를 제거하기 위해 delete 문과 drop 문이 포함되어 있다. 따라서 main 함수에서는 DB와 연결되면 먼저 20190963.txt 파일을 한 줄 씩 읽고 DB에 저장한다. 20190963.txt 파일을 모두 읽으면 type을 선택하고 그 선택에 따라 query1()부터 query5()까지의 함수를 실행하는 코드를 반복한다. 사용자가 0을 입력하면 반복문이 종료되고 20190963_del.txt 파일을 읽는다. 이 파일을 한 줄 씩 읽고 DB에 존재하는 테이블을 제거하고 나면 DB와의 연결을 종료한다.

3-2. query1()

query1() 함수는 사용자가 Type 1을 선택했을 경우에 실행된다. Type 1은 3개의 subtype이 있다. 이 함수에서는 먼저 subtype을 입력 받고 그에 해당하는 query를 실행한다.

subtype 1은 특정 트럭이 사고가 난 시점에 그 트럭에 들어있던 패키지의 고객을 찾는 query를 실행한다. 여기서 input은 트럭의 ID이고, 사고가 난 시간은 2023-06-04 12:53:49라고 가정한다. 이 때 query는 다음과 같다.

```
select customer_ID, cus_name

from ((shipment join delivery using (vehicle_ID)) join pack_order using(package_ID)) join customer using(customer_ID)

where delivery.depart_time < '2023-06-04 12:53:49' and delivery.arrive_time > '2023-06-04 12:53:49' and vehicle_ID = 've01'
```

shipment와 delivery를 vehicle_ID로 join하고, 이를 다시 pack_order와 package_ID로 join하고 customer와 customer_ID로 join한다. 이렇게 만들어진 테이블에서 사고시점이 depart_time과 arrive_time 사이에 있고, vehicle_ID가 사용자가 입력한 트럭의 ID와 일치하는 customer_ID와 cus_name을 찾는다.

subtype 2는 특정 트럭이 사고가 난 시점에 그 트럭에 들어있던 패키지의 수신자를 찾는 query를 실행한다. 여기서도 input은 트럭의 ID이고, 사고가 난 시간은 subtype1과 동일한다. 이 타입의 query는 다음과 같다.

```
select recipient_ID, rec_name

from ((shipment join delivery using (vehicle_ID)) join receive using(package_ID)) join recipient using(recipient_ID)

where delivery.depart_time < '2023-06-04 12:53:49' and delivery.arrive_time > '2023-06-04 12:53:49' and vehicle_ID = 've01'
```

전체적인 query는 subtype 1과 동일하다. 차이점은 select문인데, subtype 1과 달리, recipient_ID와 rec_name을 구한다.

subtype 3는 특정 트럭이 사고가 나기 전에 마지막으로 완료한 배송을 query를 실행한다. 여기서도 input은 트럭의 ID이고, 사고가 난 시간은 2023-06-20 12:53:49이다. 이 타입의 query는 다음과 같다.

```
select warehouse_ID, destination, depart_time, arrive_time

from delivery

where arrive_time < '2023-06-04 12:53:49' and vehicle_ID = 've01 '

 order by arrive_time desc
```

delivery 테이블에서 사고 시점이 arrive_time보다 크고 vehicle_ID가 사용자가 입력한 트럭과 동일한 결과를 arrive_time 내림차순으로 찾는다. 이 후 결과를 출력하는 코드에서 이전과 달리 모든 결과를 출력하지 않고, 첫번째 결과만 출력하면 마지막으로 완료한 배송을 얻을 수 있다.

사용자가 subtype을 선택할 때, 0을 누르면 Type을 선택하는 단계로 돌아가고, 1~3을 입력하면 결과가 출력된 후 subtype을 선택하는 단계를 반복한다.

3-3. query2()

query2() 함수는 특정 연도에 가장 많은 패키지를 보낸 고객을 찾는 함수이다. 사용자의 입력은 특정 연도를 입력하는 것이다.

```
select customer_ID, cus_name

from customer join pack_order using (customer_ID)

where order_year = year

group by customer_ID

order by count(package_ID) desc
```

customer와 pack_order를 customer_ID로 join한다. 이 테이블에서 order_year가 사용자의 입력과 동일한 결과를 찾는다. 이 결과를 customer_ID로 group화하고, 이를 package_ID의 개수의 내림차순으로 정렬한다. 이렇게 찾은 결과를 출력하는 코드에서 모든 결과를 출력하지 않고, 첫 번째 결과만 출력하면 특정 연도에 가장 많은 패키지를 보낸 고객을 얻을 수 있다.

3-4. query3()

query3() 함수는 특정 연도에 가장 많은 금액을 소비한 고객을 찾는 함수이다. 사용자는 특정 연동를 입력한다.

```
select customer_ID, cus_name

from customer join pack_order using (customer_ID)

where order_year = year

group by customer_ID

order by sum(price) desc"
```

customer와 pack_order를 customer_ID로 join한 테이블에서 order_year가 사용자가 입력한 year와 동일한 결과를 찾고 customer_ID로 그룹화한다. 이 결과를 각 그룹의 price의 합의 내림차순으로 정렬한다. 이렇게 찾은 결과를 출력하는 코드에서 모든 결과를 출력하지 않고, 첫 번째 결과만 출력하면 특정 연도에 가장 많은 금액을 소비한 고객을 얻을 수 있다.

3-5. query4()

query4() 함수는 제시간에 도착하지 못한 패키지를 구하는 함수이다. 사용자의 입력은 없으며 query는 다음과 같다.

```
select package_ID

from receive

where is_in_time = 1
```

receive 테이블에서 is_in_time의 값이 1인 package_ID를 찾는다. is_in_time은 앞서 physical schema diagram에서 설명했던 것처럼 제시간에 도착한 패키지는 0, 그렇지 않은 패키지는 1이 저장된다. 이 결과를 모두 출력하면 제시간에 도착하지 못한 패키지를 모두 구할 수 있다.

3-6. query5()

query5() 함수는 Bill을 출력하는 함수이다. Bill에는 3 종류가 있다. 사용자가 지불해야 할 비용을 보여주는 simple bill, service type에 대한 요금을 보여주는 bill, 그리고 패키지의 내용물과 각각의 요금을 보여주는 bill이다. 사용자는 1~3을 입력하여 세 종류의 bill 중 하나를 선택할 수 있고, 0을 입력하면 type을 선택하는 단계로 돌아간다. 사용자의 입력은 세 종류의 bill 모두 customer_ID와 확인하고 싶은 year와 month이다.

simple bill의 query는 다음과 같다.

```
select cus_name, address, sum(price)

from customer join pack_order using (customer_ID)

where payment_type = 'monthly' and order_year = year order_month = month and customer_ID = cus_id
```

customer와 pack_order를 customer_ID로 join한다. 이 테이블에서 payment_type이 monthly이고, order_year과 order_month가 각각 사용자가 입력한 year와 month와 동일하고, customer_ID가 사용자가 입력한 ID와 동일한 결과를 찾는다. 그 결과에서 고객의 이름과 주소, 요금의 합을 출력한다.

service type bill의 query는 다음과 같다.

```
select package_type, sum(price)

from (package join pack_order using (package_ID)) join customer using (customer_ID)

where order_year = year and order_month = month and customer_ID = cus_id;

group by package_type
```

package와 pack_order를 package_ID로 join하고, 이를 customer와 customer_ID로 join한다. 이 테이블에서 order_year와 order_month가 사용자가 입력한 year와 month와 일치하고, customer_ID가 사용자의 ID와 동일한 튜플을 찾는다. 이를 package_type으로 그룹화해서, package_type과 price의 합을 출력한다.

Itemize bill의 query는 다음과 같다.

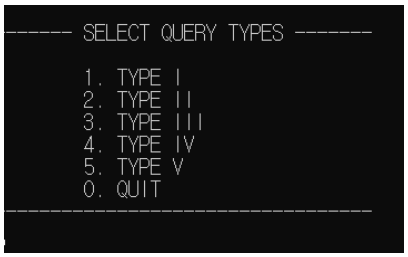
```
select content, price

from (package join pack_order using (package_ID)) join customer using (customer_ID)

where order_year = year and order_month = month and customer_ID = cus_id
```

package와 pack_order를 package_ID로 join하고, 이를 customer와 customer_ID로 join한다. 이 테이블에서 order_year와 order_month가 사용자가 입력한 year와 month와 일치하고, customer_ID가

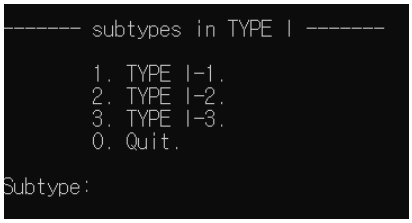
사용자의 ID와 동일한 튜플을 찾는다. 이 튜플들의 content와 price를 각각 출력한다.

4. 출력 화면

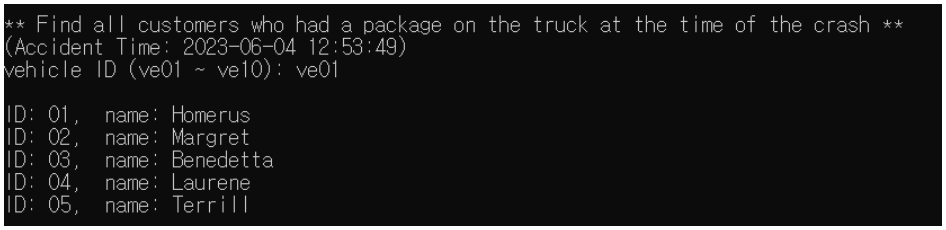
4-1. Query type 선택 장면



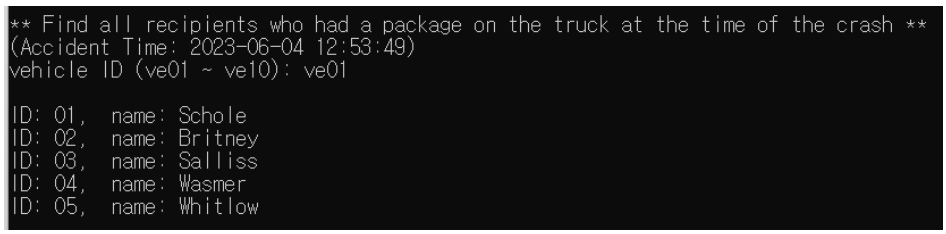
4-2. Type1의 subtype 선택 장면



4-3. Type 1-1 실행 장면 (입력: ve01)



4-4. Type 1-2 실행 장면 (입력: ve01)



4-5. Type 1-3 실행 장면 (입력: ve01)

```
** Find the last successful delivery by that truck prior to the crash **
(Accident Time: 2023-06-20 12:53:49)
vehicle ID (ve01 ~ ve10): ve01

departure: hub01,  destination: hub02
time of departure: 2023-06-04 10:53:49,  time of arrive: 2023-06-04 18:53:49
```

4-6. Type 2 실행 장면 (입력: 2023)

```
---- TYPE2 ----

** Find the customer who has shipped the most packages in certain year **
Year (2023): 2023

ID: 02,  name: Margret
```

4-7. Type 3 실행 장면 (입력: 2023)

```
---- TYPE3 ----

** Find the customer who has spent the most money on shipping in certain year **
Year (2023): 2023

ID: 02,  name: Margret
```

4-8. Type 4 실행 장면 (입력 없음)

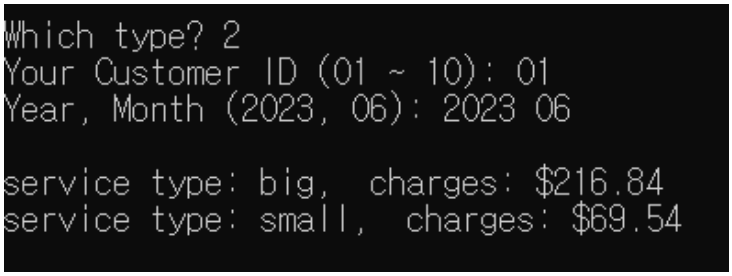
```
---- TYPE4 ----

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

Package ID: 80
Package ID: 84
Package ID: 92
Package ID: 97
```

4-9. Type 5의 Bill type 선택 장면

```
---- TYPE5 ----

---- bill type ----

1. Simple Bill
2. Service Type Bill
3. Itemize Bill
0. Quit

Which type? _
```

4-10. Type5-1(simple bill) 실행 화면 (입력: 01, 2023 06)

4-11. Type5-2(service type bill) 실행 화면 (입력: 01, 2023 06)



4-12. Type5-3(Itemize bill) 실행 화면 (입력: 01, 2023 06)