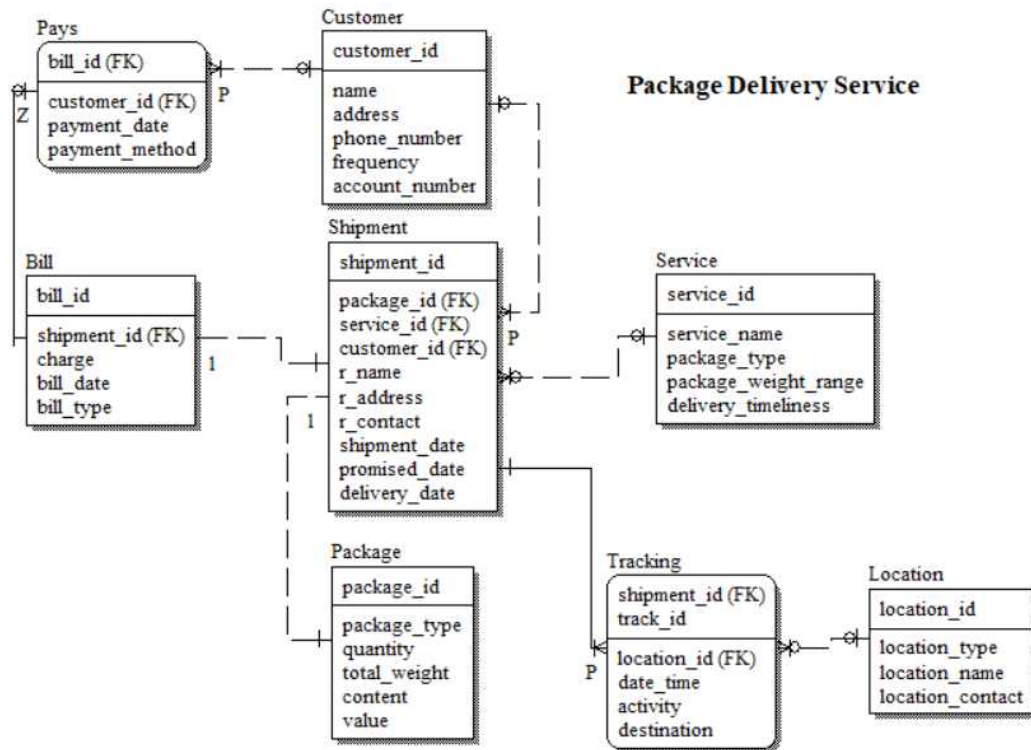


Package Delivery System DB design 제안서 2

SG Company DBA
이 주형

[Logical Schema]



I) BCNF Decomposition (simplified test)

Application Description에 따르면, logical schema의 모든 entity와 relationship에 대하여 BCNF dependency가 만족되는지 살펴라 하였기에, 우선 이 작업을 수행했습니다. 저는 기본적으로 BCNF 분해를 위해 각 relation을 살펴보고, 슈퍼키가 아닌 attribute의 그룹 α 를 기준으로 $\alpha \rightarrow \beta$ 에 해당하는 non-trivial한 FD의 존재유무를 체크하겠습니다. 이 중, 'Service' entity에 대한 simplified test 결과를 우선 제시하고 시작하겠습니다.

BCNF simplified test에선, 주어진 F 내의 $FD(\alpha \rightarrow \beta)$ 가 1) trivial한가($\beta \subseteq \alpha$) 2) α 가 슈퍼키인가를 확인하고, 둘 다 해당하지 않는 경우 Decomposition을 하게 됩니다.

'Service': Service entity에서는 다음과 같이 3개의 FD가 존재하는 F를 가집니다.

- ① service_id -> service_name, package_type, package_weight_range, delivery_timeliness
- ② service_name -> package_type, package_weight_range, delivery_timeliness
- ③ package_type, package_weight_range, delivery_timeliness -> service_name

첫 번째 FD의 좌측항 service_id의 경우, PK이므로 당연히 슈퍼키에 해당하며, 따라서 BCNF의 조건 2)를 만족하게 됩니다.

두 번째 FD의 좌측항 service_name의 경우, 역시나 그 자체로 슈퍼키에 해당합니다. service_name도 candidate key이며, 다른 속성을 모두 정의할 수 있기 때문입니다. 따라서, BCNF의 조건 2)를 만족하게 됩니다.

세 번째 FD의 좌측항 package_type, package_weight_range, delivery_timeliness 역시 세 가지의 조합에 대해 service_name과 service_id가 정의된 것이기 때문에, 슈퍼키에 해당하게 됩니다. 따라서, 이 역시 BCNF의 조건 2)를 만족하게 됩니다.

따라서, Service는 BCNF Dependency를 '만족' 한다고 할 수 있습니다.

이러한 BCNF Simplified Test를 다른 entity에도 적용해 본 결과는 다음과 같습니다.

'Customer' : PK인 customer_id 외에는, 다른 속성 간의 FD가 딱히 존재하지 않는다고 판단했습니다. name 역시 애초에 PK로 id를 설정한 이유가, 동명이인의 존재 가능성을 염두에 둔 것이기 때문에 주소나 전화번호 등 다른 요소를 결정짓지 못합니다. 따라서 Customer는 BCNF Dependency를 만족합니다.

'Bill' : PK인 shipment_id 외에는, 다른 속성 간의 FD가 딱히 보이지 않습니다. 따라서 BCNF Dependency를 만족합니다.

'Shipment': PK인 shipment_id 외에 다른 속성 간의 FD가 존재하지 않습니다. 따라서, BCNF Dependency를 만족합니다.

'Package': PK인 package_id 외에 다른 속성 간의 FD가 존재하지 않습니다. 따라서, BCNF Dependency를 만족합니다.

'Tracking': PK인 shipment_id, track_id 외에 다른 속성 간의 FD가 존재하지 않습니다. 따라서, BCNF Dependency를 만족합니다.

'Location': location_type, location_name -> location_contact 라는 FD가 존재하지만 이 경우 좌측항의 조합으로 id를 만든 것이기에 역시 슈퍼키에 해당합니다. 따라서, BCNF

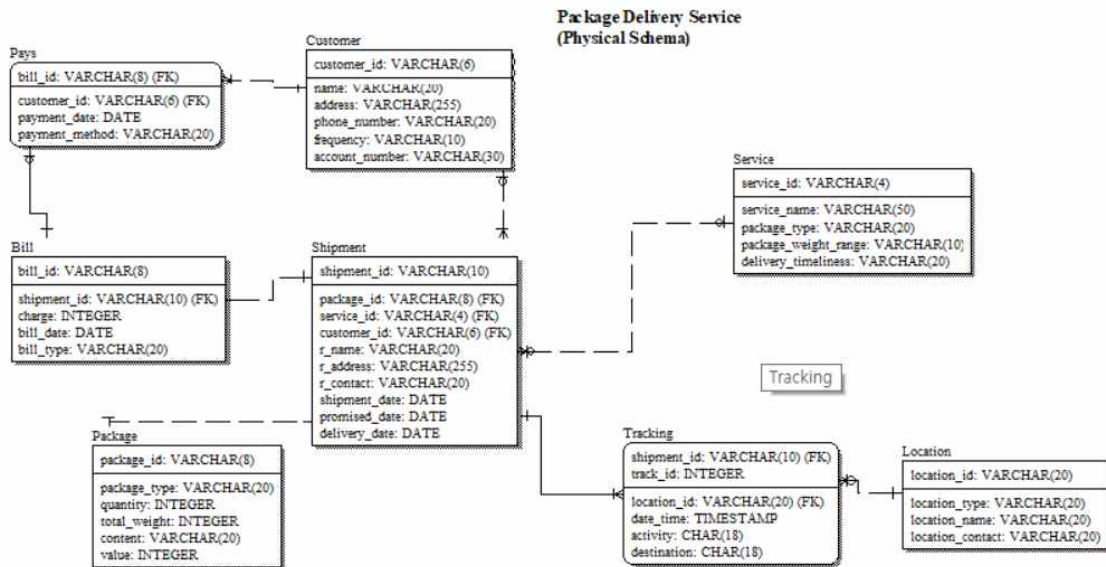
Dependency를 만족합니다.

'pays': PK인 bill_id 외에 다른 속성 간의 FD가 존재하지 않습니다. 한 명의 Customer가 여러 방식의 payment_method를 사용할 수도 있기 때문에 단순히 customer_id로는 해당 bill의 payment_method가 뭔지 알 수 없고, 두 attribute 간의 FD는 존재하지 않습니다. 따라서, BCNF Dependency를 만족합니다.

위와 같이, 현재 구성했던 logical schema의 모든 entity와 relationship은 BCNF Dependency를 만족하는 것으로 보입니다. 초기 설정 단계에서, 저희가 다양한 요소를 함께 세심히 검토했기에 가능했던 일이라 생각합니다. 결과적으로, *Decomposed Logical Schema diagram*은 이전의 *logical schema*와 동일합니다.

II-1) Physical Schema (ERWin)

university DB의 DDL을 참고로 하여, 각각의 relation에 대해 data type, domain, constraints, relationship type, allowing nulls 등을 ERWin에서 지정했습니다.



II-2) Physical Schema (SQL)

위의 ERWin에서의 지정과 constraint 등을 바탕으로, SQL에서 테이블을 생성하는 과정에서도 적용했습니다. 기본적으로 필수정보는 null값을 허용하지 않는 not null로 표기했습니다. 편의를 위해 FK가 없는 relation부터 진행했습니다.

Customer

* customer_id : 'C16670'와 같이 들어가면 되기에, varchar(6)로 설정했습니다. 그리고

PK이기에 자동적으로 not null이 됩니다.

* name : 'Joohyung Lee' 와 같이 이름이 들어가면 됩니다. 넉넉히 varchar(20)으로 설정했습니다. not null로 설정해주어 고객의 이름이 꼭 들어가게 하였습니다.

* address : 'Bojeong-ro, Giheung-gu, Yongin-si, Gyeonggi-do, Republic of Korea'와 같이 영문 주소가 들어가게 됩니다. 넉넉히 varchar(255)로 설정했습니다. 반송되는 경우를 대비해 주소 역시 not null로 설정해 꼭 들어가게 하였습니다.

* phone_number : '010-9868-1585'와 같이 전화번호가 들어가면 됩니다. 앞자리가 010이 아닌 핸드폰이나, 인터넷 전화번호 등이 들어올 것을 고려해 넉넉히 varchar(20)으로 설정해주었습니다. 연락 가능한 경우를 대비하여 not null로 설정했습니다.

* frequency : 간단히 'High, Medium, Low' 중에 하나가 들어가도록 하였기에 varchar(10)으로 설정해주었습니다. 참고용이기에 특별히 not null을 지정하진 않았습니니다.

* account_number : '110-373-036544'와 같이 계좌번호가 들어가게 했습니다. 편의를 위해 샘플 데이터에는 파트너인 신한은행의 계좌번호만 우선 고려하였는데, 만약 다른 은행의 계좌까지 포함하려면 'Shinhan 110-373-036544', 'Woori 1002-738-459298' 이런 식으로 앞에 은행명을 포함해주면 될 것 같고, 따라서 넉넉히 varchar(30)으로 설정해주었습니다. monthly_bill을 안하는 customer는 계좌가 특별히 요구되지 않으므로 not null은 지정하지 않았습니니다.

Service

모든 요소가 service의 타입을 결정짓는데 기여하기에, 모두 not null 설정하였습니다.

* service_id : 자사에는 총 15가지의 서비스를 샘플로 생성하고, SG1부터 SG15까지 매겨 id로 주었습니다. 서비스 종류가 100까지 가진 않을 것이라 생각하고, 우선 varchar(4)로 지정해주었고, PK이기에 따로 not null은 설정하지 않았습니니다.

* service_name: 각 15가지 서비스의 이름인 SG Standard Overnight F&L(Flat envelope + Light) 등이 해당됩니다. 넉넉히 varchar (50), not null을 설정해주었습니다.

* package_type: flat envelope, small box, large box로 이루어져 있습니다. 넉넉히 varchar(20), not null을 설정해주었습니다.

* total_weight_range: light 와 heavy 두 종류만 설정했습니다. 기준은 5kg을 초과하느냐, 아닌가로 구분하겠습니다. varchar(10), not null을 설정해주었습니다.

* delivery_timeliness: Overnight, Second day, Longer로 설정했습니다. varchar(20), not null을 설정해주었습니다.

Package

* package_id : P0152032 이런 식으로 들어가게 하기 위해 varchar(8)로 설정해주었고, PK이기에 따로 not null은 하지 않았습니니다.

* package_type: flat envelope, small box, large box로 구분됩니다. varchar(20), not null을 설정해주었습니다.

* quantity: 몇 개의 물건이 그 package에 포함되어있는지 나타냅니다. 수량이 100을 넘진 않을 것이니 tinyint를 사용하고, 수량이 0개보단 많은지 check 해주었습니다.

* total_weight: 그램(g)을 기준으로 하며 적당히 mediumint를 설정했습니다. 역시 0보다 크도록 설정했습니다.

- * content: 포함된 물건을 나타냅니다. varchar(20)을 설정해주고, 내용물이 위험 물품인지 확인해야하기에 not null을 설정해주었습니다.
- * value: 가액을 나타내며, 100만원 미만까지 가능하게 mediumint로 설정해주었습니다. 역시 0보다 크도록 설정했습니다.

Location

- * location_id : location_type과 location_name을 조합해서 생성한 것으로, t_1215 등에 해당합니다. varchar(20)으로 설정해주었습니다. PK이기에 따로 not null을 설정하진 않았습니다.
- * location_type : warehouse, plane, truck 등이 해당되며, varchar(20)에 not null을 설정해주었습니다.
- * location_name: 해당 location의 특정 이름이나 번호 등이 설정되며, varchar(20)에 not null을 설정해주었습니다.
- * location_contact: 해당 location의 담당 연락처에 해당하며, varchar(20)에 not null을 설정해주었습니다.

Shipment

- * shipment_id : S19-000101 의 형식으로, varchar(10)으로 지정해주었으며, PK이기에 따로 not null을 설정해주진 않았습니다.
- * package_id, service_id, customer_id : package에서 받아온 foreign key로, 동일한 도메인 설정 후 on delete set null로 설정해주었습니다. 위 3가지 정보가 사라지더라도 해당 배송 정보는 회사의 실적 및 배송전적 관리 차원에서 잔여 정보가 필요하기 때문입니다.
- * r_name: 받는 대상의 이름입니다. varchar(20), not null을 설정했습니다.
- * r_address: 받는 대상의 주소입니다. varchar(255), not null을 설정했습니다. query로 입력하는 과정에서 255글자 초과로 문제가 발생하여, insert value 과정에서는 기존에 입력되어있던 곤자가 주소(35, Baekbeom-ro, Mapo-gu, Seoul, Republic of Korea)에서 동과 호수(A110)만 남기고 일단 생략하였습니다.
- * r_contact: 받는 대상의 연락처입니다. varchar(20), not null을 설정했습니다.
- * shipment_date: 배송이 접수된 날짜입니다. DATE 타입을 사용했습니다.
- * promised_date, delivery_date: 언제까지 배송이 도착되기로 약속된 날짜, 실제 배송 날짜입니다. shipment_date보다 앞선 날짜는 아닌지 check하도록 제약을 주었으며, 역시 DATE 타입을 사용했습니다.

Bill

- * bill_id : 수많은 bill을 B0000140 형식으로 저장하기 위해 varchar(8)로 설정했습니다. PK이기에 not null은 자동적으로 반영됩니다.
- * shipment_id: Shipment에서 가져온 FK입니다. 설정 그대로 varchar(10)로 적용했습니다. 행여나 배송정보가 지워지더라도 이미 처리된, 혹은 처리되지 못한 bill은 확인해야하므로 on delete set null로 설정해주었습니다.
- * charge : 청구 금액입니다. 보통 만원 내외의 금액이 발생하므로 mediumint를 넉넉히 적용해주었습니다. 청구 금액이 0원 이상인지 check하도록 하였습니다.
- * bill_date: 청구 날짜로, DATE 타입을 사용했습니다.

* bill_type: 청구방식은 monthly, credit card, prepaid 등에 해당하게 됩니다. 넉넉히 varchar(20)으로 설정해주었습니다.

Tracking

* shipment_id : Shipment에서 가져온 foreign key로, 만약 배송 정보 자체가 사라지면 그에 대한 log 데이터는 큰 의미가 없다고 생각하여 on delete cascade를 적용해주었습니다.

* track_id: discriminator의 역할을 하며, 0부터 시작하여 차례대로 숫자가 들어가 shipment_id와 함께 PK를 구성하게 됩니다. 하나의 배송정보에 100개 이상의 log가 들어가는 경우는 없기에 tinyint로 설정해주었으며, 0보다 크도록 check를 적용해주었습니다.

* location_id: Location에서 가져온 foreign key로, 해당 location이 없어지더라도 log데이터는 유효하도록 on delete set null을 적용했습니다.

* date_time: 해당 track에 대한 날짜와 시간을 저장하도록 하였으며 DATETIME을 사용하여 "2020-01-01 10:00:00" 형식이 들어가도록 했습니다.

* activity: 해당 track에 대한 현재 배송상태나 이동, 도착 등의 세부정보를 나타내기 위해 varchar(50)으로 설정했습니다.

* destination: 해당 track에서의 목적지로, varchar(20)을 적용해주었습니다.

pays

Customer와 Bill의 관계인 pays에는 payment_date를 통해 실제 결제일을 반영하도록 * bill_id : Bill에서 가져온 foreign key로, 동일한 도메인으로 varchar(8)을 적용했습니다. bill이 삭제되는 경우, 지불정보도 함께 사라지도록 on delete cascade를 적용했습니다.

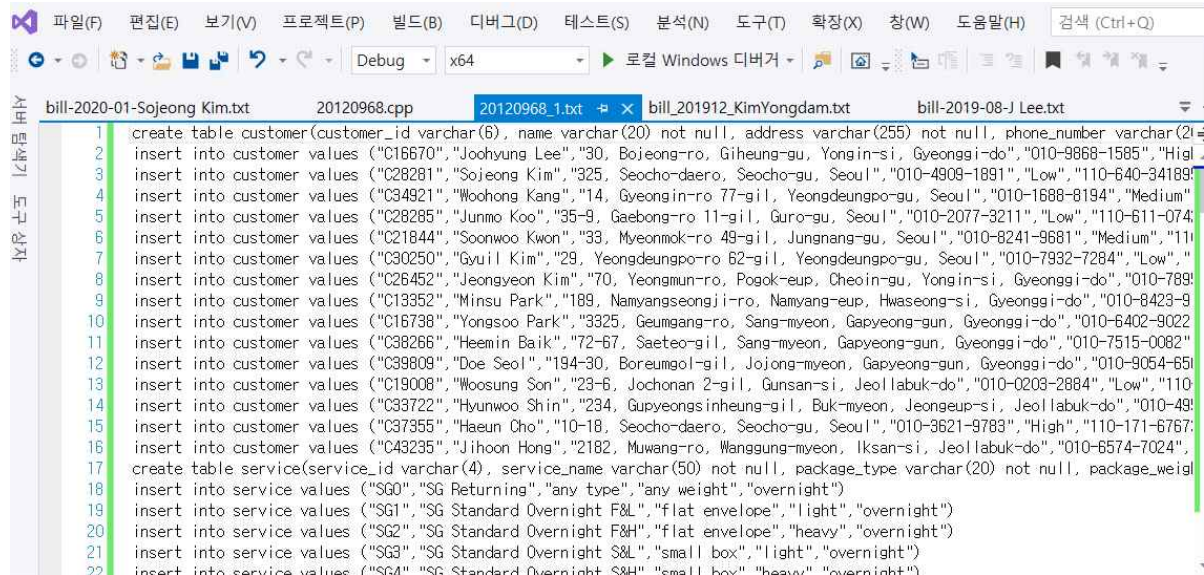
* customer_id: Customer를 참조하는 foreign key로, 동일한 도메인 varchar(6)을 적용했습니다. 고객 정보가 사라지더라도 회사 차원에서 한 해 수익 등을 계산할 때 pays를 사용할 수 있도록 on delete set null로 적용했습니다.

* payment_date: 청구된 bill에 대한 지불이 이루어진 날로, DATE 타입을 적용했습니다.

* payment_method: 청구된 bill에 대한 실제 지불 방식으로, credit card, cash, bank transfer 등에 해당합니다. varchar(20)을 적용했습니다. 아직 지불하지 않은 경우를 대비해 not null은 적용하지 않았습니다.

III) ODBC 처리

a)txt로 CRUD 하기

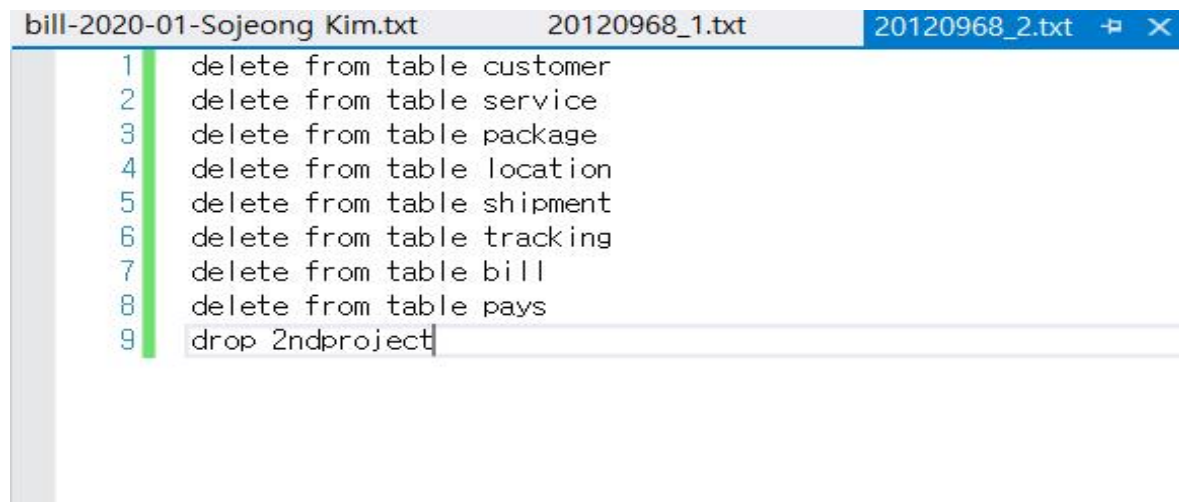


```
1 create table customer(customer_id varchar(6), name varchar(20) not null, address varchar(255) not null, phone_number varchar(20) not null, email varchar(26) not null, gender varchar(2) not null, job_id varchar(3) not null, salary number(8,2) not null, commission_pct number(3,2) not null, manager_id varchar(6) not null, department_id number(4) not null, primary key(customer_id))
2 insert into customer values ('C16670','Joohyung Lee','30, Bojeong-ro, Giheung-gu, Yongin-si, Gyeonggi-do','010-9868-1585','High', 'M', 'SALES', 12000.00, 0.10, null, null, 'SALES')
3 insert into customer values ('C28281','Sojeong Kim','325, Seocho-daero, Seocho-gu, Seoul','010-4909-1891','Low','110-640-34189','M','SALES', 12000.00, 0.10, null, null, 'SALES')
4 insert into customer values ('C34921','Woohong Kang','14, Gyeongin-ro 77-gil, Yeongdeungpo-gu, Seoul','010-1688-8194','Medium','110-640-34189','M','SALES', 12000.00, 0.10, null, null, 'SALES')
5 insert into customer values ('C28285','Junmo Koo','35-9, Gaebong-ro 11-gil, Guro-gu, Seoul','010-2077-3211','Low','110-611-074','M','SALES', 12000.00, 0.10, null, null, 'SALES')
6 insert into customer values ('C21844','Soonwoo Kwon','33, Myeonmok-ro 49-gil, Jungnang-gu, Seoul','010-8241-9681','Medium','110-640-34189','M','SALES', 12000.00, 0.10, null, null, 'SALES')
7 insert into customer values ('C30250','Gyuil Kim','29, Yeongdeungpo-ro 62-gil, Yeongdeungpo-gu, Seoul','010-7932-7284','Low','110-640-34189','M','SALES', 12000.00, 0.10, null, null, 'SALES')
8 insert into customer values ('C26452','Jeongyeon Kim','70, Yeongmun-ro, Pogok-eup, Cheoin-gu, Yongin-si, Gyeonggi-do','010-789','M','SALES', 12000.00, 0.10, null, null, 'SALES')
9 insert into customer values ('C13352','Minsu Park','189, Namyangseongji-ro, Namyang-eup, Hwaseong-si, Gyeonggi-do','010-8423-9','M','SALES', 12000.00, 0.10, null, null, 'SALES')
10 insert into customer values ('C16738','Yongsu Park','3325, Geumgang-ro, Sang-myeon, Gapyeong-gun, Gyeonggi-do','010-6402-9022','M','SALES', 12000.00, 0.10, null, null, 'SALES')
11 insert into customer values ('C38266','Heemin Baik','72-67, Saeteo-gil, Sang-myeon, Gapyeong-gun, Gyeonggi-do','010-7515-0082','M','SALES', 12000.00, 0.10, null, null, 'SALES')
12 insert into customer values ('C39809','Doe Seol','194-30, Boreumgol-gil, Jojong-myeon, Gapyeong-gun, Gyeonggi-do','010-9054-65','M','SALES', 12000.00, 0.10, null, null, 'SALES')
13 insert into customer values ('C19008','Woosung Son','23-6, Jochonari 2-gil, Gunsan-si, Jeollabuk-do','010-0203-2884','Low','110-640-34189','M','SALES', 12000.00, 0.10, null, null, 'SALES')
14 insert into customer values ('C33722','Hyunwoo Shin','234, Gupyeongsinheung-gil, Buk-myeon, Jeongeup-si, Jeollabuk-do','010-49','M','SALES', 12000.00, 0.10, null, null, 'SALES')
15 insert into customer values ('C37355','Haeun Cho','10-18, Seocho-daero, Seocho-gu, Seoul','010-3621-9783','High','110-171-6767','M','SALES', 12000.00, 0.10, null, null, 'SALES')
16 insert into customer values ('C43235','Jihoon Hong','2182, Muwang-ro, Wanggung-myeon, Iksan-si, Jeollabuk-do','010-6574-7024','M','SALES', 12000.00, 0.10, null, null, 'SALES')
17 create table service(service_id varchar(4), service_name varchar(50) not null, package_type varchar(20) not null, package_weight varchar(20) not null, primary key(service_id))
18 insert into service values ('SG0','SG Returning','any type','any weight','overnight')
19 insert into service values ('SG1','SG Standard Overnight F&L','flat envelope','light','overnight')
20 insert into service values ('SG2','SG Standard Overnight F&H','flat envelope','heavy','overnight')
21 insert into service values ('SG3','SG Standard Overnight S&L','small box','light','overnight')
22 insert into service values ('SG4','SG Standard Overnight S&H','small box','heavy','overnight')
```

명세서의 요구사항대로 txt파일로 CRUD가 가능하도록 했습니다.

20120968_1.txt의 경우, table을 CREATE하고, value들을 INSERT하는 역할을 합니다.

20120968_2.txt의 경우, table을 drop하는 역할을 합니다.



```
1 delete from table customer
2 delete from table service
3 delete from table package
4 delete from table location
5 delete from table shipment
6 delete from table tracking
7 delete from table bill
8 delete from table pays
9 drop 2ndproject
```

b) Query 처리

명세서에 주어진 총 7가지의 쿼리를 처리했습니다.

TYPE 1은 명세서의 가정 상에서 truck 1721에 대한 것만 처리했습니다. 하지만 다른 location이 붕괴될 경우도 대비해 location_contact를 "destroyed"로 설정했고, 이를 활용한 처리도 나중에 가능하도록 했습니다.

```
0. done
Which type of query? 1
----- TYPE 1 -----
Input the number of truck : 1366
Truck 1366 is not destroyed.

----- TYPE 1 -----
Input the number of truck : 1721

----- Subtypes in TYPE 1 -----
1. TYPE 1-1
2. TYPE 1-2
3. TYPE 1-3
Which type of query? 1
----- TYPE 1-1 -----
** Find all customers who had a package on the truck at the time of the crash. **
Customer ID: C37355 | C16670 | C28281 |

----- Subtypes in TYPE 1 -----
1. TYPE 1-1
2. TYPE 1-2
3. TYPE 1-3
Which type of query? 2
----- TYPE 1-2 -----
** Find all recipients who had a package on the truck at the time of the crash. **
Customer name: Jisoo Park | Suji Jeong | Jisoo Park |

----- Subtypes in TYPE 1 -----
1. TYPE 1-1
2. TYPE 1-2
3. TYPE 1-3
Which type of query? 3
----- TYPE 1-3 -----
** Find the last successful delivery by that truck prior to the crash. **
Last Successful Delivery on: 2019-01-21 10:00:00

----- Subtypes in TYPE 1 -----
1. TYPE 1-1
2. TYPE 1-2
3. TYPE 1-3
Which type of query? 0
```


TYPE 1,2,3의 경우, 동명이인을 고려해 Name 대신 ID를 쓰도록 처리했습니다.

```
----- SELECT QUERY TYPES -----
1. TYPE I
2. TYPE II
3. TYPE III
4. TYPE IV
5. TYPE V
0. QUIT
Which type of query? 2

----- TYPE II -----
** Find the customer who has shipped the most packages in certain year **
Which Year? : 2020
Customer ID: C34921 shipped most in 2020
** Find the customer who has shipped the most packages in certain year **
Which Year? : 2019
Customer ID: C16670 shipped most in 2019
** Find the customer who has shipped the most packages in certain year **
Which Year? : 0

----- SELECT QUERY TYPES -----
1. TYPE I
2. TYPE II
3. TYPE III
4. TYPE IV
5. TYPE V
0. QUIT
Which type of query? 3

----- TYPE III -----
** Find the customer who has spent the most money on shipping in the past certian year **
Which Year? : 2020
Customer ID: C34921 spent most in 2020
** Find the customer who has spent the most money on shipping in the past certian year **
Which Year? : 2019
Customer ID: C16670 spent most in 2019
** Find the customer who has spent the most money on shipping in the past certian year **
Which Year? : 0

----- SELECT QUERY TYPES -----
```

TYPE4의 경우, shipment relation에서 약속날짜(promised_date)와 실제배송일(delivery_date)를 함께 가지고 있기에, 비교하는 구문으로 단순히 처리했습니다.

```
----- SELECT QUERY TYPES -----
1. TYPE I
2. TYPE II
3. TYPE III
4. TYPE IV
5. TYPE V
0. QUIT
Which type of query? 4

----- TYPE IV -----
** Find those packages that were not delivered within the promised time**
P0213111 | P0324157 | P1574142 | P1620328 | P1630211 |

----- 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 **
Customer Name : Sojeong Kim
Which month(YYYY-MM)? 2020-01
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

----- QUIT -----
Have a nice day!!!

C:\Users\Joobe\source\repos\mysql\wx64\Debug\mysql.exe(프로세스 2268개)이(가) 종료되었습
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 종
이 창을 닫으려면 아무 키나 누르세요...
```

TYPE 5의 경우, 하나의 텍스트 파일에 3가지의 bill이 다 나타나도록 다음과 같이 처리했습니다. 이 때, 텍스트 파일명은 bill-날짜-이름.txt가 됩니다.

bill-2020-01-Sojeong Kim.txt		20120968.cpp	20120968_1.txt	bill_201912_KimYongdam.txt	bill-2019-08-J Lee.txt
1	*** Simple Billing List ***				
2		Customer Name	Address		Amount
3		Sojeong Kim	325, Seocho-daero, Seocho-gu, Seoul		5000
4					
5					
6	*** Type of Service Billing List ***				
7		Bill_date	Type_of_Service	Charge	
8		2020-01-15	SG1	0	
9		2020-01-19	SG3	5000	
10		2020-01-21	SG3	0	
11					
12					
13	*** Itemized Billing List ***				
14		Bill_id	Content	Value	Service_id Bill_type Bill_date
15		B0000491	SG Company Resume	1000	SG7 credit card 2019-02-03
16		B0012901	Love Letter	10000	SG1 prepaid 2020-01-15
17		B0013107	IPhone	999000	SG3 credit card 2020-01-19
18		B0013205	IPhone	999000	SG3 returning 2020-01-21
19					

기본적으로, 무한반복 루프는 while(1)을 사용하였습니다.

또한 Bill의 경우 view를 이용해 보다 간소히 만들었습니다.

추후 다른 형태에 Bill도 필요하시다면, joobe24@sogang.ac.kr로 문의 부탁드립니다.)