과목명: 데이터베이스시스템

<project #2>

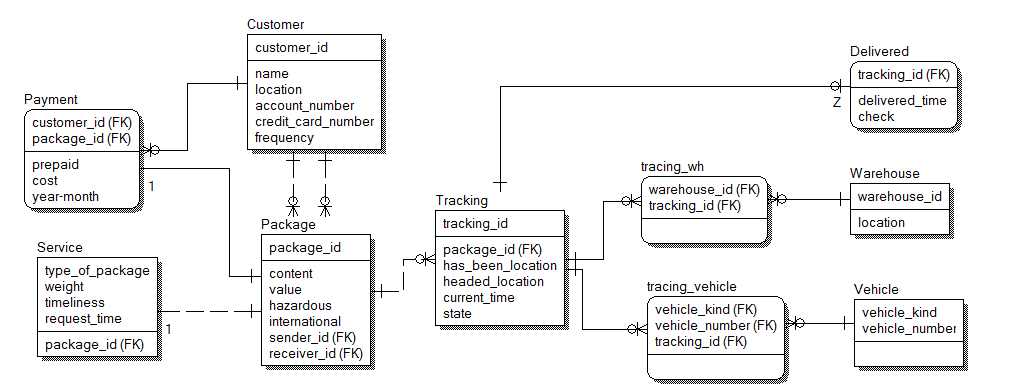
서강대학교 수학과

20141362

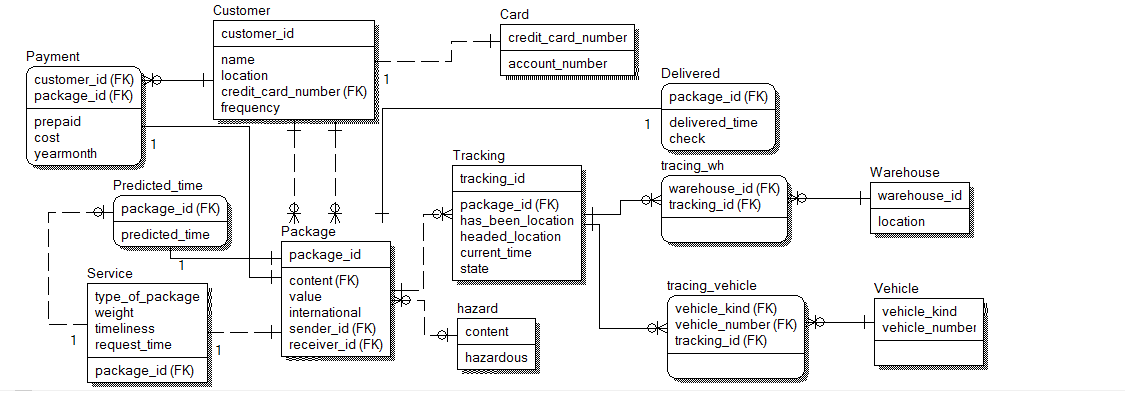
천지환

1. **Logical schema 변경 사항**

* 변경 전 logical schema



* 변경 후 logical schema



* 변경사항
  1. logical schema 기본 구조 및 attriute명 변경

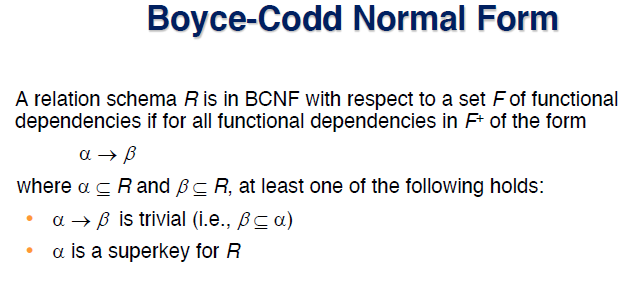
기존의 payment table의 year\_month attribute의 이름을 yearmonth로 바꾸었습니다. Mysql 내장 함수에 year\_month라는 이름이 존재해 year\_month를 사용할 수 없기에 형태를 바꾸었습니다.

그리고 기존의 delivered의 tracking\_id를 package\_id로 바꿔주고 delivered와 tracking 사이에 있던 relation을 없애고 delivered와 package 사이에 relation을 만들어 주었습니다. 원래 tracking 마다 배송이 완료되었는지 확인 하는 역할을 하는 것이 delivered table이였습니다. 하지만 2차 과제를 하며 생각해본 결과 package 하나에 대해서만 배송완료가 되었는지 확인하면 되는데 package 하나에 여러 개 연결되어 있는 tracking을 대상으로 일일히 배송완료 를 체크해주는 것은 불필요한 중복이라 생각해 바꿔주었습니다.

추가로 기존에 없었던 Predicted\_time(package\_id, predicted\_time) table을 만들어 주었습니다. 이는 원래 있던 Service table에서 request\_time과 timeliness를 이용하여 예상 도착시간을 구할 수 있다고 생각해 따로 table이나 attribute를 두지 않은 상태였었는데, 이번 프로젝트에서 query문을 짜면서 mysql 내에서 if문과 함수를 통시에 사용하는데 어려움이 있어 따로 table을 구성해주었습니다. Predicted\_time table은 Package table의 package\_id를 foreign key로 받아서 primary key로 가지고 있으며 attribute로 predicted\_time을 가지고 있습니다. 그리고 Package table과 one-to-one의 cardinality를 갖고 Service table과도 one-to-one cardinality를 갖습니다.

* 1. bcmf decompose

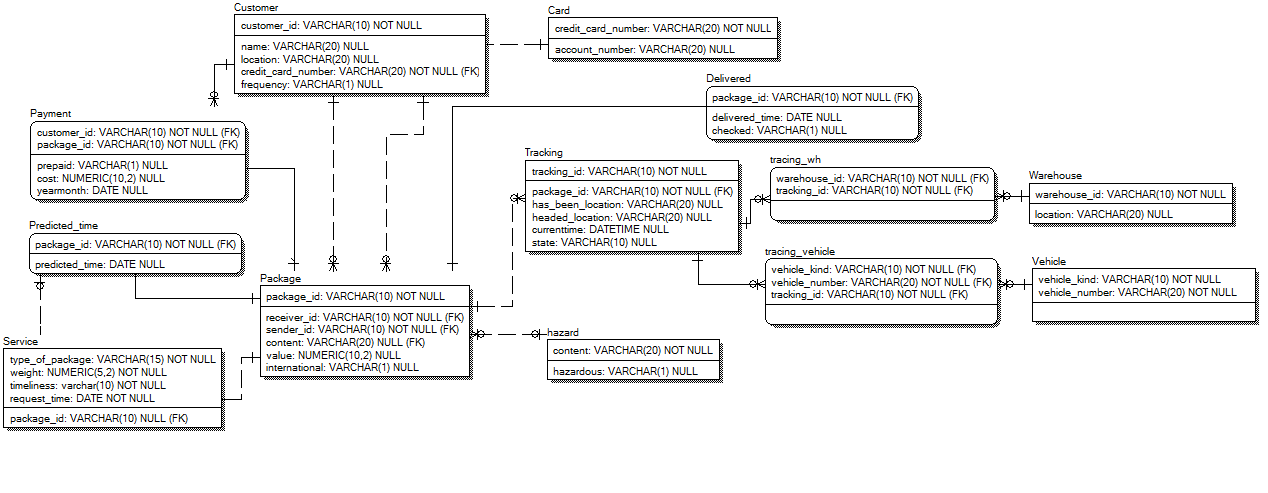
bcnf의 기본 조건은 아래 사진과 같습니다.



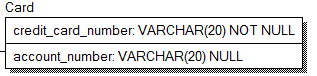
이 조건에 맞게 decompose를 하기 위해서 기존의 logical schema를 보면 Customer table에서 신용카드 당 하나의 계좌에 연결되어 있기에 credit\_card\_number->account\_number의 functional dependancy가 존재합니다. 근데 credit\_card\_number은 Customer table에서 primary key도 아니고 trivial한 functional dependency도 아니기에 기존 Customer table에서 (account\_number) – (credit\_card\_number) 인 account\_number을 제거하고 Card(credit\_card\_number, account\_number) table을 새롭게 생성하였습니다. 그 뒤 one-to-one의 cardinality를 갖도록 설정하였습니다.

또한 기존에 있던 Package table에서 content 와 hazardous를 보면 content의 종류에 따라 위험한지 아닌지 둘 중 한가지로 정해지므로 content값 1개에 여러 hazardous가 matching 될 일이 없습니다. 따라서 content -> hazardous 라는 functional dependency를 갖습니다. 이 때 content 또한 primary key가 아니고 이 functional dependency가 trivial하지도 않기에 Package table에서 (hazardous) – (content) 인 hazardous를 지워주고 hazardous(content, hazardous) table을 새로 만들어 주었습니다. 이 table은 Package table과 one-to-one cardinality를 갖습니다.

1. **Physical schema diagram**



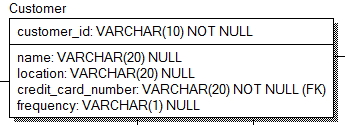
* 1. Card



Credit\_card\_number은 card table의 primary key를 맡고있는 attribute입니다. 보통 카드번호는 16자리로 알고있지만 외국의 경우 다를 수도 있다고 생각했기에 20자리를 설정하고 varchar()을 선택했습니다. 이는 card의 primary key역할이기에 not null이 설정했습니다.

Account\_number는 고객의 계좌 번호를 저장하는 attribute입니다. 정기 결제 고객의 경우 account\_number로도 결제할수 있기에 이를 attribute로 만들어 주었습니다. 계좌 번호의 경우 한국에서도 제각각이고 외국의 경우 또 어떨지 몰라 넉넉히 크기 20의 varchar()을 설정했습니다.

2.2 Customer



Customer\_id는 customer table의 Customer table을 식별하는 primary key 역할을 하는 attribute입니다. 현재 customer은 15개의 tuple만을 갖지만 실제 배송회사에서는 customer의 수가 무척 많아질 수 있기에 10 의 크기를 주었습니다. 또한 id가 꼭 숫자의 형태인 것만은 아니기에 varchar()을 선택했습니다. Primary key 이기에 not null 을 설정했습니다.

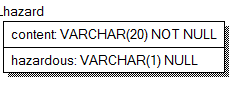
Name은 고객이름을 저장하는 attribute의 경우 한국인의 경우 길어봤다 5글자 정도라고 생각할 수 있지만 명세서의 기본이 영어이고 외국인의 이름은 생각보다 길어질수 있다고 생각하기에 20의 크기를 주었고 이름은 기본 글자형태이기고 길이가 제각각이기에 varchar()을 선택했습니다.

Location은 customer가 사는 지역을 저장하는 attribute입니다. 사는 지역 또한 길어질수 있기에 20의 크기 설정을, 길이가 모두 제각각이기에 varchar()을 선택했습니다.

Credit\_card\_number은 customer의 카드 번호를 저장하는 attribute입니다. 보통 카드번호는 16자리로 알고있지만 외국의 경우 다를 수도 있다고 생각했기에 20자리를 설정하고 varchar()을 선택하였습니다. 이는 card의 pk역할이기에 foreign key 이고 not null이 설정했습니다..

Frequency는 정기 결제 고객인지를 나타내는 attribute로 y or n의 값을 저장 하기 위해 varchar(1)을 설정했습니다.

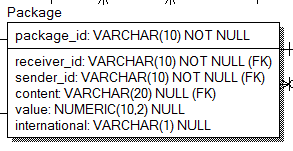
2.3 hazard



Content는 hazard table의 primary key 역할을 하는 attribute입니다. Primary key 역할을 수행하기에 not null이 설정하였습니다. Content는 배송하는 물품이 무엇인지 저장하는 attribute로써 제품명이 다양할 수 있다는 생각에 크기 20 종류 varchar()을 설정했습니다.

Hazardous는 해당 content가 hazardous material 인지 체크하는 attribute로 y or n 값을 갖도록 설정하기 위해 크기 1 종류 varchar()을 설정했습니다.

2.4 Package



Package\_id는 package table을 식별하는 primary key 역할을 하는 attribute입니다. Primary key 역할을 하기에 not null이며 이번 프로젝트에서는 tuple개수가 15개 밖에 안되지만 실제로는 값의 종류가 무척 크고 다양할 수 있기에 크기 10을 종류는 varchar()을 설정했습니다.

Receiver\_id는 package를 수령하는 customer의 id를 나타내는 attribute입니다. Cutomer table의 customer\_id를 foreign key로 가지고 있으며 그 datatype과 constraint를 그대로 가져왔습니다.

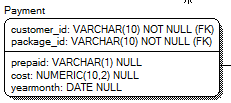
Sender\_id는 package를 발송하는 customer의 id를 나타내는 attribute입니다. Customer table의 customer\_id를 foreign key로 가지고 있고 그 datatype과 constraint를 그대로 가져왔습니다.

Content는 package의 내용물이 무엇인지 나타내는 attribute입니다. 이는 위험 물질이나 international 배송일 경우 알 필요가 있으며 hazard table의 content를 foreign key로 가지고 있으며 그 datatype과 constraint를 그대로 가져왔습니다.

Value는 international 배송일 경우 해당 content의 값어치를 알 필요가 있기에 설정한 attribute입니다. Value는 기본적으로 숫자값을 갖기에 numeric을 설정해주었고 물건의 값어치는 천차만별인 점에 정수 10자리, 소수점은 2자리까지는 표시할 필요가 있다고 생각하기에 numeric(10,2)를 설정해주었습니다.

International은 package가 해외배송인지 아닌지 식별해주는 attribute입니다. 이는 y or n 의 값을 갖도록 하게 하기 위하여 varchar(1)을 설정하였습니다.

2.5 Payment



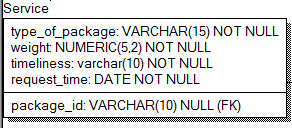
Customer\_id는 customer table의 customer\_id를 foreign key로, package\_id는 package table의 package\_id를 foreign key로 갖고 datatype과 constraint는 모두 동일합니다. 이 둘이 같이 primary key 역할을 합니다.

Prepaid는 해당 package가 반송 제품인지 알기 위한 attribute입니다. 반송 제품의 경우 prepaid가 되어있고 그렇지 않을경우 되어있지 않기에 이는 y or n 의 값을 갖게 하기 위해 varchar(1)을 설정했습니다.

Cost의 경우 배송 비용을 저장하는 attribute입니다. 비용이기에 numeric을 설정 해주었고 범위를 넉넉히 주기 위해 정수 10자리를 주었습니다. 그리고 소수점의 경우 까지 표시하기 위해 소수점을 2자리 설정하여 numeric(10,2)를 설정했습니다.

Yearmonth는 해당 payment가 언제 배송한 package에 대한 payment인지 저장하기 위한 attribute입니다. 배송한 당시의 시간 값을 갖고, 시간값이기에 datatype이 date로 year-month-day 값을 갖습니다.

2.6 Service



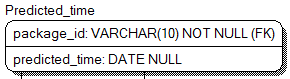
Type\_of\_package는 flat\_envelope, small\_box, larger\_boxes의 값 중 하나를 갖는 attribute로 package의 크기에 관한 attribute입니다. 이 세개의 값을 받기 위해 varchar(15)를 설정했습니다.

Weight는 package의 무게에 대한 attribute로 숫자값을 같기에 numeric을 설정해주었습니다. 무게는 범위를 넉넉히 하기위해 정수값을 5자리, 가벼운 package경우 세밀히 측정하기위해 소수점 2자리까지 설정해주었다. 따라서 numeric(5,2)를 갖습니다.

Timeliness는 overnight, second\_day, third\_day, week, longer 중 하나의 값을 갖는 attribute로 배송기간에 대한 attribute이다. 이를 받기 위해 10자리의 varchar()을 설정했습니다.

Request\_time은 배송을 요청한 시간으로 해당 package가 시간내에 배송되었는지를 확인하기 위한 predicted\_time을 계산하기 위해 넣어준 attribute입니다. 배송을 요청한 일시를 저장하는 attribute로 datatype은 date(yyyy-mm-dd)를 갖습니다.

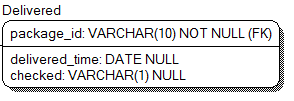
2.7 Predicted\_time



Package\_id는 package table의 package\_id 를 foreign key로 갖고 datatype과 constraint를 그대로 갖습니다.

Predicted\_time은 service의 request\_time과 timeliness로 계산할수 잇는 derived attribute로 timeliness가 overnight일 경우 request\_time에 하루를, second\_day일 경우 이틀을, third\_day일 경우 3일을, week일 경우 7일을 더해준 값입니다. 일시 형태의 데이터를 갖기에 datatype은 date입니다.

2.8 Delivered

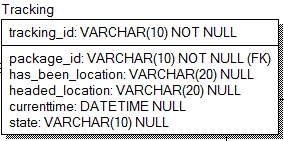


Package\_id는 package table의 package\_id 를 foreign key로 갖고 datatype과 constraint를 그대로 갖습니다.

Delivered\_time은 배송이 완료된 시간을 저장하는 attribute로 일시 data 형식을 갖기에 date를 datatype으로 갖습니다.

Checked 는 해당 배송이 완료된 것을 customer가 확인했는지를 나타내는 attribute로 y or n 의 값을 갖게 하게 위해 varchar(1)을 설정하였습니다.

2.9 Tracking



Tracking\_id는 tracking table을 식별하는 역할을 하는 primary key입니다. Primary key이기에 not null이며 현재의 tuple은 적지만 실제 데이터의 경우 무척 많은 값을 가지기에 10자리를 갖는 varchar()을 설정해 주었습니다.

Package\_id는 package table의 package\_id 를 foreign key로 갖고 datatype과 constraint를 그대로 갖습니다.

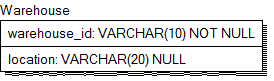
Has\_been\_location은 마지막으로 도착한 지역을 저장하는 attribute입니다. 지역명은 상당히 길 수 있기에 20자리를 갖는 varchar()을 설정했습니다.

Headed\_location은 다음 이동할 지역을 저장하는 attribute입니다. 지역명은 상당히 길 수 있기에 20자리를 갖는 varchar()을 설정했습니다.

Currenttime은 tracking시 위치가 바뀔떄마다 도착시간을 알기 위한 attribute입니다. 일시 뿐만 아니라 시간까지 포함한 값을 갖기에 date를 datatime type으로 갖습니다.

State는 현재 package의 상태가 배송 준비중인지, 배송 시작인지, 이동중인지, warehouse인지 완료인지를 알기 위한 attribute로 ready, start, moving, warehouse, done 중 하나의 값을 갖고 이 값을 받기 위해 varchar(10)을 설정하였습니다.

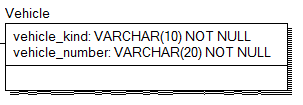
2.10 Warehouse



Warehouse\_id는 warehouse table을 식별하기 위한 attribute로 primary key입니다. Primary key이기에 not null이며 warehouse는 실제로 무척 다양할 수 있기에 10자리의 varchar을 설정했습니다.

Location은 warehouse가 있는 위치를 나타내는 attribute로 앞선 location들과 마찬가지로 지역명은 길수 있기에 20자리를 갖는 varchar()을 설정해주었습니다.

2.11 Vehicle

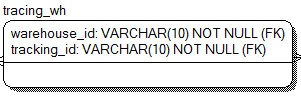


Vehicle\_kind는 운송 수단의 종류를 저장하는 attribute로 airplane, truck, train, vessel, boat 등의 값을 갖습니다. 이를 저장하기 위해 varchar(10)를 설정해주었고 primary key의 성분중 하나이기에 not null을 설정했습니다.

Vehicle\_number는 해당 운송 수단의 번호(ex 차량 번호)를 갖는 attribute입니다. 운송수단에 따라 번호가 다양할수 있다고 생각해 varchar(20)을 설정하였고 primary key의 성분중 하나이기에 not null을 설정했습니다.

둘을 합쳐 primary key로 설정한 것은 kind가 달라고 같은 번호가 있을 수도 있다고 생각했기때문입니다.

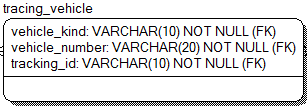
2.12 tracing\_wh(relation entity)



warehouse\_id는 warehouse table의 warehouse\_id 를 foreign key로 갖고 datatype과 constraint를 그대로 갖습니다.

tracking\_id는 tarcking table의 tracking\_id 를 foreign key로 갖고 datatype과 constraint를 그대로 갖습니다.

2.13 tracing\_vehicle(relation table)



Vehicle\_kind는 vehicle table의 tracking\_kind 를 foreign key로 갖고 datatype과 constraint를 그대로 갖습니다.

Vehicle\_number는 vehicle table의 vehicle\_number 를 foreign key로 갖고 datatype과 constraint를 그대로 갖습니다.

tracking\_id는 tarcking table의 tracking\_id 를 foreign key로 갖고 datatype과 constraint를 그대로 갖습니다.

1. **Mysql Query**

* **TYPE 1-1 (truck 1721이 2020-06-29 14:00:00에 destroyed 되었다고 가정)**

query 설명 : 먼저 tracking table에서 package\_id로 grouping을 한다음 해당 package의 가장 마지막 tracking\_id 값을 attribute로 갖는 track\_recent(package\_id, tracking\_id) table을 만듭니다. 이렇게 하면 해당 package의 마지막 tracking정보를 가지고 있는 tracking\_id값을 알 수 있습니다.

그 뒤 track\_recent table과 tracing\_vehicle table을 natural join 한뒤 vehicle\_kind가 truck이고 vehicle\_number가 1721인 package\_id 값들을 갖는 crush\_package(package\_id) table을 만듭니다. Track\_recent와 tracing\_vehicle을 natural join할 경우 두 테이블의 tracking\_id 값이 같은 package만 남게됩니다. 이렇게 되면 가장 마지막 tracking이 1721번 truck 이었던 package\_id 값들을 갖게 됩니다. 가장 마지막 tracking이 1721번 트럭이라는 것은 truck이 crash될 당시 그 트럭에 있던 package라는 뜻입니다.

그 뒤 package table을 이용하여 package\_id가 crush\_package(package\_id)값에 속한 sender\_id를 구하면 1721번 트럭이 crash될 당시의 package를 갖는 sender 들의 id를 얻을수 있습니다. 이를 sender(customer\_id)로 만듭니다.

마지막으로 이 sender들의 id를 이용해 해당 sender들의 이름을 구하기 위해 customer와 sender를 natural join한 뒤 name과 customer\_id를 select합니다.

with track\_recent(package\_id, tracking\_id) as

(select package\_id, max(tracking\_id)

from tracking

group by package\_id

),

crush\_package(package\_id) as

(select package\_id

from track\_recent natural join tracing\_vehicle

where vehicle\_kind = 'truck' and vehicle\_number = 1721

),

sender(customer\_id) as

(select sender\_id

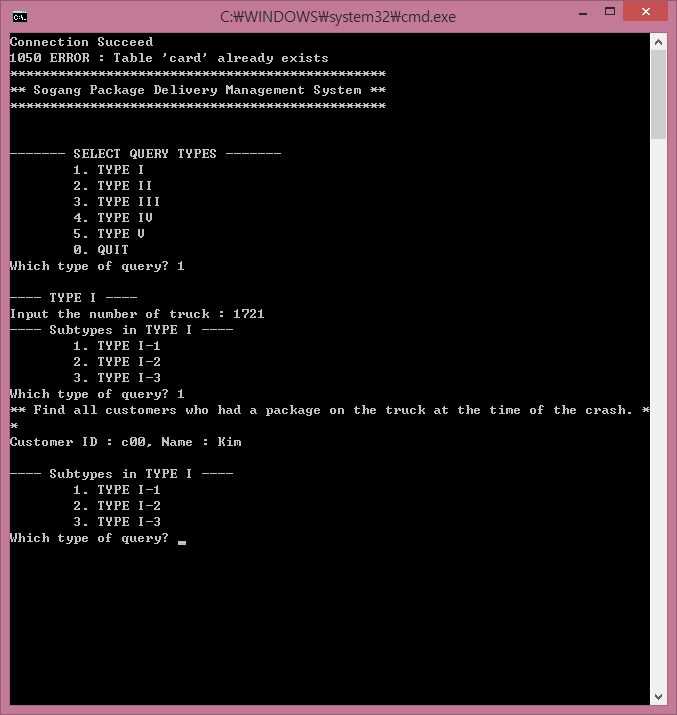
from package

where package.package\_id in (select package\_id from crush\_package)

)

select distinct customer\_id, name

from customer natural join sender



* **TYPE 1-2(truck 1721이 2020-06-29 14:00:00에 destroyed 되었다고 가정)**

query 설명: type 1-1의 과정과 유사합니다. Sender가 receiver로 바뀐다는 차이만 있습니다.

먼저 tracking table에서 package\_id로 grouping을 한다음 해당 package의 가장 마지막 tracking\_id 값을 attribute로 갖는 track\_recent(package\_id, tracking\_id) table을 만듭니다. 이렇게 하면 해당 package의 마지막 tracking정보를 가지고 있는 tracking\_id값을 알 수 있습니다.

그 뒤 track\_recent table과 tracing\_vehicle table을 natural join 한뒤 vehicle\_kind가 truck이고 vehicle\_number가 1721인 package\_id 값들을 갖는 crush\_package(package\_id) table을 만듭니다. Track\_recent와 tracing\_vehicle을 natural join할 경우 두 테이블의 tracking\_id 값이 같은 package만 남게됩니다. 이렇게 되면 가장 마지막 tracking이 1721번 truck 이었던 package\_id 값들을 갖게 됩니다. 가장 마지막 tracking이 1721번 트럭이라는 것은 truck이 crash될 당시 그 트럭에 있던 package라는 뜻입니다.

그 뒤 package table을 이용하여 package\_id가 crush\_package(package\_id)값에 속한 sender\_id를 구하면 1721번 트럭이 crash될 당시의 package를 갖는 receiver 들의 id를 얻을수 있습니다. 이를 receiver(customer\_id)로 만듭니다.

마지막으로 이 receiver들의 id를 이용해 해당 receiver들의 이름을 구하기 위해 customer와 receiver를 natural join한 뒤 name과 customer\_id를 select합니다.

with track\_recent(package\_id, tracking\_id) as

(select package\_id, max(tracking\_id)

from tracking

group by package\_id

),

crush\_package(package\_id) as

(select package\_id

from track\_recent natural join tracing\_vehicle

),

receiver(customer\_id) as

(select receiver\_id

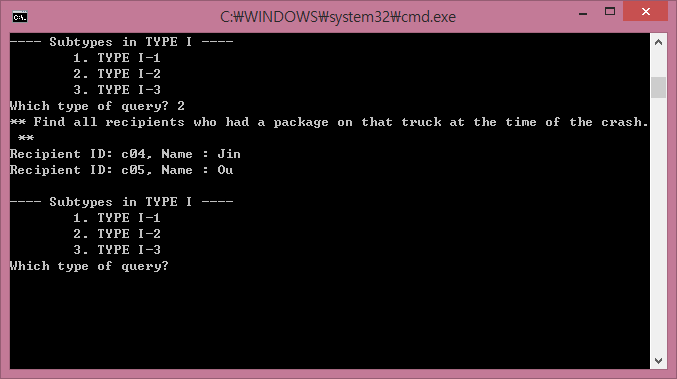
from package

where package.package\_id in (select package\_id from crush\_package)

)

select customer\_id, name

from customer natural join receiver



* **TYPE 1-3(truck 1721이 2020-06-29 14:00:00에 destroyed 되었다고 가정, 배송완료는 warehouse에 도착하는 것도 포함(그 트럭에서 내리는 순간 배송 완료))**

query 설명 : 먼저 tracking과 tracing\_vehicle을 natural join한 뒤 1721 truck을 쓴적 잇는 package\_id와 tracking\_id를 모두 찾아 use\_truck1721(package\_id, tracking\_id) table로 만듭니다.

그 뒤 tracking table에서 package\_id로 grouping을 한다음 해당 package의 가장 마지막 tracking\_id 값을 attribute로 갖는 track\_recent(package\_id, tracking\_id) table을 만듭니다. 이렇게 하면 해당 package의 마지막 tracking정보를 가지고 있는 tracking\_id값을 알 수 있습니다.

그리고 use\_truck1721로부터 tracking\_id가 track\_recent에 없는 package들을 뽑아 success\_pid(package\_id) table을 만듭니다. 이렇게 되면 1721번 truck을 사용한 적은 있고 현재는 사용 중이지 않은 즉 truck에서 배송을 완료한 package\_id 들의 값을 얻게됩니다.

이를 이용해 package table로부터 package\_id 값이 success\_pid의 table에 속해있는 content와 package\_id를 select해줍니다. 이러면 1721번 트럭을 사용했고 사고 전에 배송 완료했던 package들의 content를 구할 수 있습니다.

with use\_truck1721(package\_id, tracking\_id) as

(select package\_id, tracking\_id

from tracking natural join tracing\_vehicle

where vehicle\_kind = ‘truck’ and vehicle\_number = '1721'

),

track\_recent(package\_id, tracking\_id) as

(select package\_id, max(tracking\_id)

from tracking

group by package\_id

),

success\_pid(package\_id) as

(select distinct package\_id

from use\_truck1721

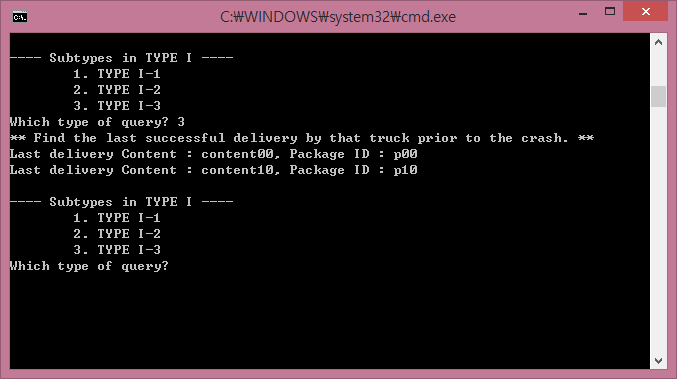
where tracking\_id not in (select distinct tracking\_id from track\_recent)

)

select content, package\_id

from package

where package\_id in (select package\_id from success\_pid)



* **TYPE 2(receiver측에서 배송비를 내 sender측에서는 배송비가 없다고 가정 c00가 sender, c01~c14가 receiver. 따라서 가장 많이 주문한사람과 가장 비용이 큰사람이 다를 수 있음.)(아래의 query문에서는 입력받은 연도가 2020이라 가정)**

query 설명 : payment table로부터 customer\_id와 yearmonth attribute중 year값만 추출 해 grouping을 해줍니다. 그 뒤 이 group들의 package 수를 count해 package\_num\_year(customer\_id, year, num) table을 만들어줍니다. 이렇게 할 경우 해당 고객의 연도별 주문 수를 알 수 있습니다.

그 뒤 package\_num\_year table로부터 배송 횟수가 가장 많았던 고객을 뽑아서 max\_customer(customer\_id, max\_num) table을 만듭니다. where절의 num = (select max(num) from package\_num\_year where year = 2020)) 조건을 이용하면 num이 2020년의 최대값인 경우만 뽑아줄 수 있습니다.

마지막으로 customer table로부터 customer\_id가 max\_customertable에 속해 있는 customer의 이름과 id을 select해주면 해당년도에 주문수가 최대인 고객의 이름과 id를 구할 수 있습니다.

with package\_num\_year(customer\_id, year, num) as

(select customer\_id, year(yearmonth), count(package\_id)

from payment

group by customer\_id, year(yearmonth)),

max\_customer(customer\_id, max\_num) as

(select customer\_id, num

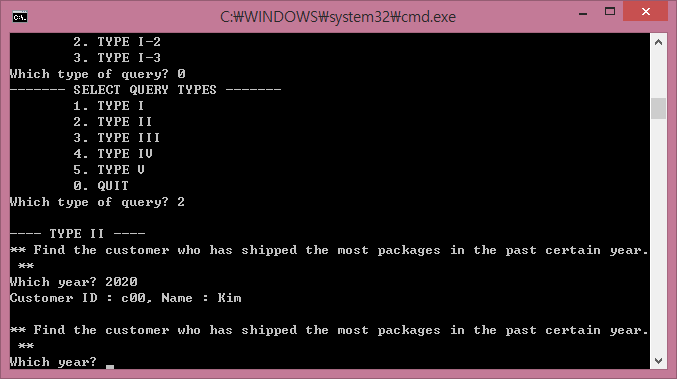
from package\_num\_year

where year = 2020 and num = (select max(num) from package\_num\_year where year = 2020))

select customer\_id, name

from customer

where customer\_id in (select customer\_id from max\_customer)



* **TYPE 3(receiver측에서 배송비를 내 sender측에서는 배송비가 없다고 가정 c00가 sender, c01~c14가 receiver. 따라서 가장 많이 주문한사람과 가장 비용이 큰사람이 다를 수 있음.) (아래의 query문에서는 입력받은 연도가 2020이라 가정)**

query 설명 : type2와 과정이 유사합니다.

먼저 payment table로부터 customer\_id와 yearmonth attribute중 year값만 추출 해 grouping을 해줍니다. 그 뒤 이 group들의 cost를 sum해서 package\_sum\_year(customer\_id, year, sum) table을 만들어줍니다. 이렇게 할 경우 해당 고객의 연도별 주문비용을 알 수 있습니다.

그 뒤 package\_sum\_year table로부터 주문 비용이 가장 컸던 고객을 뽑아서 customer\_sum\_year(customer\_id, sum) table을 만듭니다. where sum = (select max(sum) from package\_sum\_year where year = 2020)) 조건을 이용하면 2020년 비용의 총합인 sum이 2020년의 최대값인 경우만 뽑아줄 수 있습니다.

마지막으로 customer table로부터 customer\_id가 customer\_sum\_year 에 속해 있는 customer의 이름과 id을 select해주면 해당년도에 주문 비용이가 최대인 고객의 이름과 id를 구할 수 있습니다.

with package\_sum\_year(customer\_id, year, sum) as

(select customer\_id, year(yearmonth), sum(cost)

from payment

group by customer\_id, year(yearmonth)),

customer\_sum\_year(customer\_id, sum) as

(select customer\_id, sum

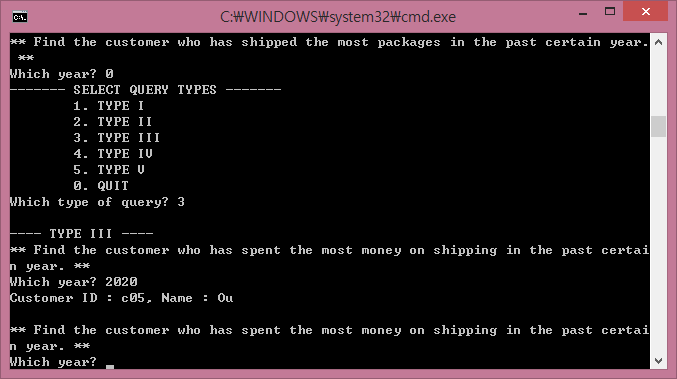
from package\_sum\_year

where year = 2020 and sum = (select max(sum) from package\_sum\_year where year = 2020))

select customer\_id, name

from customer

where customer\_id in (select customer\_id from customer\_sum\_year)



* **TYPE 4**

query 설명 : 먼저 predicted\_time table과 deliver table을 natural join한 뒤 배송 완료 시일이 예상 시일 보다 큰 package\_id들을 뽑아서 late\_package(package\_id)을 만들어 줍니다.

그 뒤 package table로부터 package\_id가 late\_package에 속한 package들을 뽑아 package\_id와 content를 select합니다. 이렇게 하면 배송도착기간이 약속기간보다 늦은 package들의 id와 content들을 구할 수 있습니다.

with late\_package(package\_id) as

(select package\_id

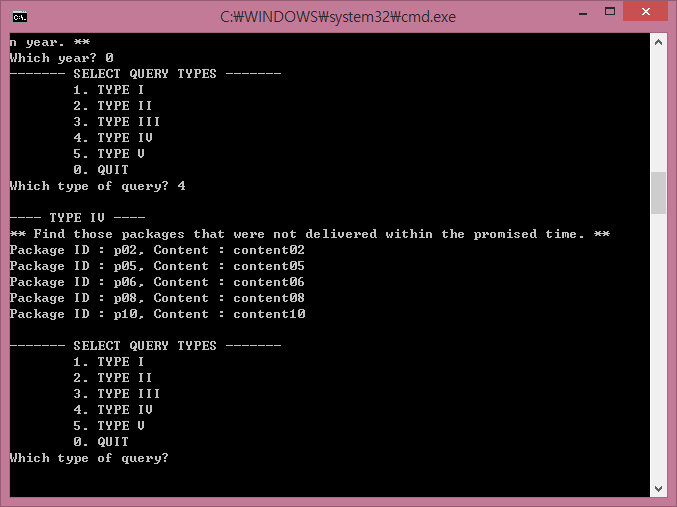
from predicted\_time natural join delivered

where predicted\_time < delivered\_time)

select package\_id, content

from package

where package\_id in (select package\_id from late\_package)



* **TYPE 5(tuple의 year과 month가 다 제각각 떨어져있어서 이름과 연도에 따라서 bill에 데이터 값이 아예 없거나 1개 밖에 없는 경우가 대다수 Ou, 2020-06의 경우 2개) (아래의 query문에서는 입력받은 일자가 2020-06, 이름이 Ou라고 가정)**

query1 설명 : 먼저 payment table로부터 customer\_id와 yearmonth attribute에서 year-month 형태로 date를 뽑아 grouping 합니다. 그뒤 cost의 sum을 구해 cost\_sum(customer\_id, ym, sum\_cost) table을 만들어줍니다. 이렇게하면 고객 id의 월별 배송비용의 총합을 구할 수 있습니다. 그 뒤 cost\_sum과 customer을 join해서 입력받은 이름과 연도-월에 해당하는 값들을 select해줍니다. 이때 출력 양식을 의해 location을 추가로 select해 customer\_id, name, location, sum\_cost를 select해줍니다. 이렇게 할 경우 입력받은 고객의 이름과 월에 해당하는 비용의 총합과 지역, 이름, 고객 id를 알 수 있습니다.

with cost\_sum(customer\_id, ym, sum\_cost) as

(select customer\_id, date\_format(yearmonth, '%Y-%m'), sum(cost)

from payment

group by customer\_id, date\_format(yearmonth, '%Y-%m'))

select customer\_id, name, location, sum\_cost

from customer join cost\_sum using(customer\_id)

where ym = '2020-06' and customer.name = 'Ou'

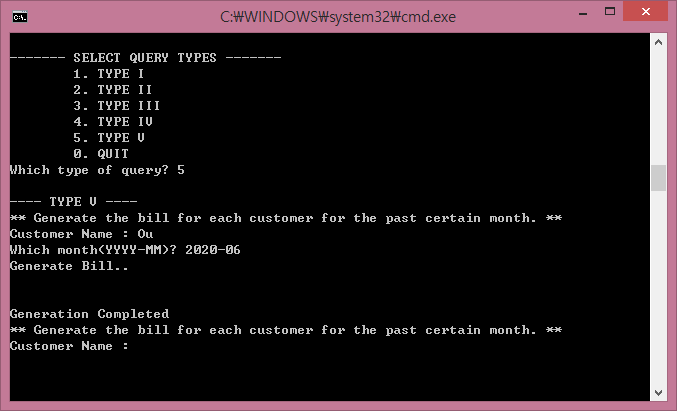
-query2 설명 : bill에서 요구하는 attribute가 다양하고 이는 여러 table에 나눠서 존재하기에 다수의 table들을 join을 이용해 합친뒤 입력받은 월과 이름에 해당하는 bill 정보들을 select합니다.

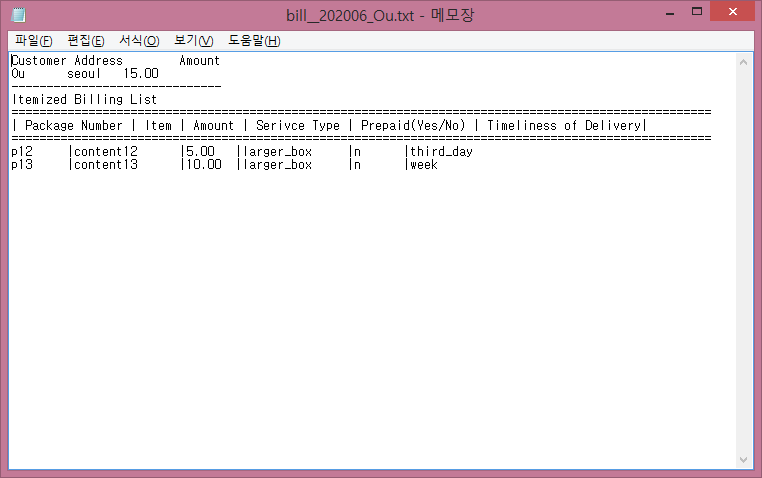
Package\_id와 content를 얻기위해 package table이, cost, prepaid를 얻기 위해 payment table이 type\_of\_package, timeliness를 얻기 위해 service table, 이름을 where절에 조건으로 걸기위해 customer table이 필요합니다.

select package\_id, content, cost, type\_of\_package, prepaid, timeliness

from ((package join payment using(package\_id)) join service using(package\_id)) join customer using(customer\_id)

where date\_format(yearmonth, '%Y-%m') = '2020-06' and name = 'Ou'





1. **C API code(CRUD에서 시간이 좀 걸립니다! 컴파일하고 조금만 기다려주세요!)**

**(c언어에서는 앞서 설명한 mysql query와 table을 이용한 ui를 만들뿐이므로 주석으로 간단히 설명하였습니다.)**

#include <stdio.h>

#include <stdlib.h>

#include "mysql.h"

#pragma comment(lib, "libmysql.lib")

const char\* host = "localhost";

const char\* user = "root";

const char\* pw = "ghksdl86..";

const char\* db = "db\_proj2";

int ui(MYSQL\* connection, MYSQL\* conn\_); //functino for user interface

int pre\_sql(MYSQL\* connection, MYSQL\* conn\_); //function for create table and insert data

int del\_sql(MYSQL\* connection, MYSQL\* conn\_); //functino for delete and drop table

int main(void) {

MYSQL\* connection = NULL;

MYSQL conn;

MYSQL\_RES\* sql\_result;

MYSQL\_ROW sql\_row;

if (mysql\_init(&conn) == NULL)

printf("mysql\_init() error!");

connection = mysql\_real\_connect(&conn, host, user, pw, db, 3306, (const char\*)NULL, 0);

if (connection == NULL)

{

printf("%d ERROR : %s\n", mysql\_errno(&conn), mysql\_error(&conn));

return 1;

}

else

{

printf("Connection Succeed\n");

if (mysql\_select\_db(&conn, db))

{

printf("%d ERROR : %s\n", mysql\_errno(&conn), mysql\_error(&conn));

return 1;

}

pre\_sql(connection, &conn);

ui(connection, &conn);

del\_sql(connection, &conn);

mysql\_close(connection);

}

return 0;

}

int ui(MYSQL\* connection, MYSQL\* conn\_) {

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\* Sogang Package Delivery Management System \*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n\n");

//while for select type

while (1) {

MYSQL\_RES\* sql\_result;

int type\_num;

MYSQL\_ROW sql\_row;

printf("------- SELECT QUERY TYPES -------\n");

printf("\t1. TYPE I\n");

printf("\t2. TYPE II\n");

printf("\t3. TYPE III\n");

printf("\t4. TYPE IV\n");

printf("\t5. TYPE V\n");

printf("\t0. QUIT\n");

printf("Which type of query? ");

scanf("%d", &type\_num);

if (type\_num > 5 || type\_num < 0) {

printf("TYPE number error! please choose 0~5\n");

}

if (type\_num == 0) {

printf("HAVE A GOOD DAY :)\n");

return 0;

}

else if (type\_num == 1) {

int truck\_num;

int destroy\_flag = 1;

printf("\n---- TYPE I ----\n");

//get truck number

while (1) {

printf("Input the number of truck : ");

scanf("%d", &truck\_num);

if (truck\_num == 0) break; // if get 0 go to select

//-------------code for get destroy\_Flag

//if truck is not destroyed get truck number until get0 or destroyed truck number

if (truck\_num != 1721) {

printf("Truck %d is not destroyed.\n\n", truck\_num);

}

else {

break;

}

}

if (truck\_num != 0) {

//while for select subtype

while (1) {

int sub\_num;

printf("---- Subtypes in TYPE I ----\n");

printf("\t1. TYPE I-1\n");

printf("\t2. TYPE I-2\n");

printf("\t3. TYPE I-3\n");

printf("Which type of query? ");

scanf("%d", &sub\_num);

//if get 0 then go to select

if (sub\_num == 0) break;

if (sub\_num > 3 || sub\_num < 0) {

printf("Wrong sub type number!\n");

}

if (sub\_num == 1) {

printf("\*\* Find all customers who had a package on the truck at the time of the crash. \*\*\n");

const char\* query = "with track\_recent(package\_id, tracking\_id) as\

(select package\_id, max(tracking\_id)\

from tracking\

group by package\_id\

),\

crush\_package(package\_id) as\

(select package\_id\

from track\_recent natural join tracing\_vehicle\

where vehicle\_kind = \'truck\' and vehicle\_number = 1721\

),\

sender(customer\_id) as\

(select sender\_id\

from package\

where package.package\_id in(select package\_id from crush\_package)\

)\

select distinct customer\_id, name\

from customer natural join sender";

int state = 0;

state = mysql\_query(connection, query);

if (state != 0)

{

printf("%d ERROR : %s\n", mysql\_errno(conn\_), mysql\_error(conn\_));

return 1;

}

if (state == 0)

{

sql\_result = mysql\_store\_result(connection);

while ((sql\_row = mysql\_fetch\_row(sql\_result)) != NULL)

{

printf("Customer ID : %s, Name : %s\n", sql\_row[0], sql\_row[1]);

}

printf("\n");

mysql\_free\_result(sql\_result);

}

}

else if (sub\_num == 2) {

printf("\*\* Find all recipients who had a package on that truck at the time of the crash. \*\*\n");

const char\* query = "with track\_recent(package\_id, tracking\_id) as\

(select package\_id, max(tracking\_id)\

from tracking\

group by package\_id\

),\

crush\_package(package\_id) as\

(select package\_id\

from track\_recent natural join tracing\_vehicle\

),\

receiver(customer\_id) as\

(select receiver\_id\

from package\

where package.package\_id in(select package\_id from crush\_package)\

)\

select customer\_id, name\

from customer natural join receiver";

int state = 0;

state = mysql\_query(connection, query);

if (state != 0)

{

printf("%d ERROR : %s\n", mysql\_errno(conn\_), mysql\_error(conn\_));

return 1;

}

if (state == 0)

{

sql\_result = mysql\_store\_result(connection);

while ((sql\_row = mysql\_fetch\_row(sql\_result)) != NULL)

{

printf("Recipient ID: %s, Name : %s\n", sql\_row[0], sql\_row[1]);

}

printf("\n");

mysql\_free\_result(sql\_result);

}

}

else if (sub\_num == 3) {

printf("\*\* Find the last successful delivery by that truck prior to the crash. \*\*\n");

const char\* query = "with use\_truck1721(package\_id, tracking\_id) as\

(select package\_id, tracking\_id\

from tracking natural join tracing\_vehicle\

where vehicle\_kind = \'truck\' and vehicle\_number = 1721\

),\

track\_recent(package\_id, tracking\_id) as\

(select package\_id, max(tracking\_id)\

from tracking\

group by package\_id\

),\

success\_pid(package\_id) as\

(select distinct package\_id\

from use\_truck1721\

where tracking\_id not in(select distinct tracking\_id from track\_recent)\

)\

select content, package\_id\

from package\

where package\_id in(select package\_id from success\_pid)";

int state = 0;

state = mysql\_query(connection, query);

if (state != 0)

{

printf("%d ERROR : %s\n", mysql\_errno(conn\_), mysql\_error(conn\_));

return 1;

}

if (state == 0)

{

sql\_result = mysql\_store\_result(connection);

while ((sql\_row = mysql\_fetch\_row(sql\_result)) != NULL)

{

printf("Last delivery Content : %s, Package ID : %s\n", sql\_row[0], sql\_row[1]);

}

printf("\n");

mysql\_free\_result(sql\_result);

}

}

}

}

}

else if (type\_num == 2) {

printf("\n---- TYPE II ----\n");

while (1) {

char year[5];

printf("\*\* Find the customer who has shipped the most packages in the past certain year. \*\*\n");

printf("Which year? ");

scanf("%s", year);

//if get 0 then go to select

if (!strcmp(year,"0")) {

break;

}

char query[1024] = "with package\_num\_year(customer\_id, year, num) as (select customer\_id, year(yearmonth), count(package\_id) from payment group by customer\_id, year(yearmonth)), max\_customer(customer\_id, max\_num) as (select customer\_id, num from package\_num\_year where year =";

strcat(query, year);

strcat(query, " and num = (select max(num) from package\_num\_year where year = ");

strcat(query, year);

strcat(query, ")) select customer\_id, name from customer where customer\_id in(select customer\_id from max\_customer)");

int state = 0;

state = mysql\_query(connection, query);

if (state != 0)

{

printf("%d ERROR : %s\n", mysql\_errno(conn\_), mysql\_error(conn\_));

return 1;

}

if (state == 0)

{

sql\_result = mysql\_store\_result(connection);

while ((sql\_row = mysql\_fetch\_row(sql\_result)) != NULL)

{

printf("Customer ID : %s, Name : %s\n", sql\_row[0], sql\_row[1]);

}

printf("\n");

mysql\_free\_result(sql\_result);

}

}

}

else if (type\_num == 3) {

printf("\n---- TYPE III ----\n");

while (1) {

int year;

printf("\*\* Find the customer who has spent the most money on shipping in the past certain year. \*\*\n");

printf("Which year? ");

scanf("%d", &year);

//if get 0 then go to select

if (year == 0) {

break;

}

const char\* query = "with package\_sum\_year(customer\_id, year, sum) as\

(select customer\_id, year(yearmonth), sum(cost)\

from payment\

group by customer\_id, year(yearmonth)),\

customer\_sum\_year(customer\_id, sum) as\

(select customer\_id, sum\

from package\_sum\_year\

where year = 2020 and sum = (select max(sum) from package\_sum\_year where year = 2020))\

select customer\_id, name\

from customer\

where customer\_id in(select customer\_id from customer\_sum\_year)";

int state = 0;

state = mysql\_query(connection, query);

if (state != 0)

{

printf("%d ERROR : %s\n", mysql\_errno(conn\_), mysql\_error(conn\_));

return 1;

}

if (state == 0)

{

sql\_result = mysql\_store\_result(connection);

while ((sql\_row = mysql\_fetch\_row(sql\_result)) != NULL)

{

printf("Customer ID : %s, Name : %s\n", sql\_row[0], sql\_row[1]);

}

printf("\n");

mysql\_free\_result(sql\_result);

}

}

}

else if (type\_num == 4) {

printf("\n---- TYPE IV ----\n");

printf("\*\* Find those packages that were not delivered within the promised time. \*\*\n");

const char\* query = "with late\_package(package\_id) as\

(select package\_id\

from predicted\_time natural join delivered\

where predicted\_time < delivered\_time)\

select package\_id, content\

from package\

where package\_id in(select package\_id from late\_package)";

int state = 0;

state = mysql\_query(connection, query);

if (state != 0)

{

printf("%d ERROR : %s\n", mysql\_errno(conn\_), mysql\_error(conn\_));

return 1;

}

if (state == 0)

{

sql\_result = mysql\_store\_result(connection);

while ((sql\_row = mysql\_fetch\_row(sql\_result)) != NULL)

{

printf("Package ID : %s, Content : %s\n", sql\_row[0], sql\_row[1]);

}

printf("\n");

mysql\_free\_result(sql\_result);

}

}

else if (type\_num == 5) {

printf("\n---- TYPE V ----\n");

while (1) {

char name[20];

char ym[10];

char year[10];

char month[10];

char output\_filename[50] = "bill\_";

char \*ptr;

printf("\*\* Generate the bill for each customer for the past certain month. \*\*\n");

printf("Customer Name : ");

scanf("%s", name);

if (!strcmp(name, "0")) {

break;

}

printf("Which month(YYYY-MM)? ");

scanf("%s", ym);

if (!strcmp(ym, "0")) {

break;

}

printf("Generate Bill..\n");

//tokenize year and month

ptr = strtok(ym, "-");

strcpy(year, ptr);

ptr = strtok(NULL, "-");

strcpy(month, ptr);

//txt file format

/\*

Customer Address Amount

Kim Yongdam AS816 $3000

------------------------------

Itemized Billing List

====================================================================================================

| Package Number | Item | Amount | Serivce Type | Payment Type | Timeliness of Delivery|

====================================================================================================

P00000011 | airPod Pro | $350 | Flat Envelope | Credit Card | Overnight

P03002111 | iPhone 11 Pro, MG, 256GB | $1400 | Small Box | Prepaid(reserved) | Second day

P10222935 | Macbook Air, RG, i5/16GB/512GB | $0 | Large Box | Returned(malfunctioning) | Returned

P10222935 | Macbook Air, RG, i5/16GB/512GB | $1250 | Large Box | 5 days(delayed)

----------------------------------------------------------------------------------------------------

\*/

//make output file name

strcat(output\_filename, "\_");

strcat(output\_filename, year);

strcat(output\_filename, month);

strcat(output\_filename, "\_");

strcat(output\_filename, name);

strcat(output\_filename, ".txt");

//remake ym

strcat(ym, "-");

strcat(ym, month);

//write bill on setting output file name

FILE \* o\_fp = fopen(output\_filename, "w");

fprintf(o\_fp, "Customer Address\tAmount\n");

//------------------------------------------------query

char query1[1024] = "with cost\_sum(customer\_id, ym, sum\_cost) as (select customer\_id, date\_format(yearmonth, \'%Y-%m\'), sum(cost) from payment group by customer\_id, date\_format(yearmonth, \'%Y-%m\')) select customer\_id, name, location, sum\_cost from customer join cost\_sum using(customer\_id) where ym = \'";

strcat(query1, ym);

strcat(query1, "\' and customer.name = \'");

strcat(query1, name);

strcat(query1, "\'");

int state = 0;

state = mysql\_query(connection, query1);

if (state != 0)

{

printf("%d ERROR : %s\n", mysql\_errno(conn\_), mysql\_error(conn\_));

return 1;

}

if (state == 0)

{

sql\_result = mysql\_store\_result(connection);

while ((sql\_row = mysql\_fetch\_row(sql\_result)) != NULL)

{

fprintf(o\_fp, "%s\t%s\t%s\n", name, sql\_row[2], sql\_row[3]);

}

printf("\n");

mysql\_free\_result(sql\_result);

}

fprintf(o\_fp, "------------------------------\n");

fprintf(o\_fp, "Itemized Billing List\n");

fprintf(o\_fp, "====================================================================================================\n");

fprintf(o\_fp, "| Package Number | Item | Amount | Serivce Type | Prepaid(Yes/No) | Timeliness of Delivery|\n");

fprintf(o\_fp, "====================================================================================================\n");

//------------------------------------------------query

char query2[256] = "select package\_id, content, cost, type\_of\_package, prepaid, timeliness from((package join payment using(package\_id)) join service using(package\_id)) join customer using(customer\_id) where date\_format(yearmonth, \'%Y-%m\') = \'";

strcat(query2, ym);

strcat(query2, "\' and name = \'");

strcat(query2, name);

strcat(query2, "\'");

state = 0;

state = mysql\_query(connection, query2);

if (state != 0)

{

printf("%d ERROR : %s\n", mysql\_errno(conn\_), mysql\_error(conn\_));

return 1;

}

if (state == 0)

{

sql\_result = mysql\_store\_result(connection);

while ((sql\_row = mysql\_fetch\_row(sql\_result)) != NULL)

{

fprintf(o\_fp, "%s\t|%s\t|%s\t|%s\t|%s\t|%s\t\n", sql\_row[0], sql\_row[1], sql\_row[2], sql\_row[3], sql\_row[4], sql\_row[5]);

}

printf("\n");

mysql\_free\_result(sql\_result);

}

fclose(o\_fp);

printf("Generation Completed\n");

}

}

}

}

//function for create table and insert data

int pre\_sql(MYSQL\* connection, MYSQL\* conn\_) {

char query[1023];

int state = 0;

FILE\* fp = fopen("20141362\_1.txt", "r");//txtfile for reate table and insert data

if (!fp) {

printf("No such file!\n");

return -1;

}

while (!feof(fp)) {

fgets(query, sizeof(query), fp);

query[strlen(query) - 1] = '\0';

//printf("%s\n",query);

state = mysql\_query(connection, query);

if (state != 0)

{

printf("%d ERROR : %s\n", mysql\_errno(conn\_), mysql\_error(conn\_));

return -1;

}

}

fclose(fp);

return 0;

}

//functino for delete and drop table

int del\_sql(MYSQL\* connection, MYSQL\* conn\_) {

char query[1023];

int state = 0;

FILE\* fp = fopen("20141362\_2.txt", "r");//txtfile for reate table and insert data

if (!fp) {

printf("No such file!\n");

return -1;

}

while (!feof(fp)) {

fgets(query, sizeof(query), fp);

query[strlen(query) - 1] = '\0';

//printf("%s\n",query);

state = mysql\_query(connection, query);

if (state != 0)

{

printf("%d ERROR : %s\n", mysql\_errno(conn\_), mysql\_error(conn\_));

return -1;

}

}

fclose(fp);

return 0;

}