데이터베이스시스템
PROJECT2

심리학과

20190345 김동현

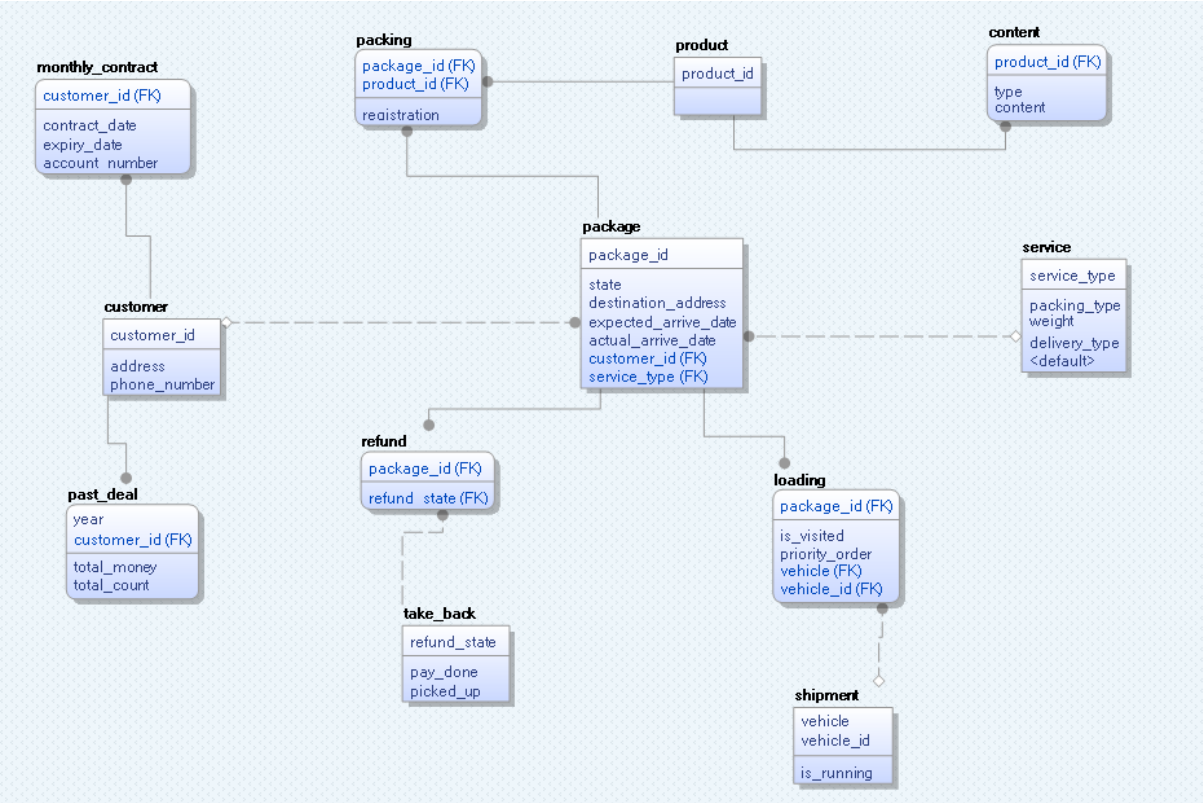
목차

1. 프로젝트 소개

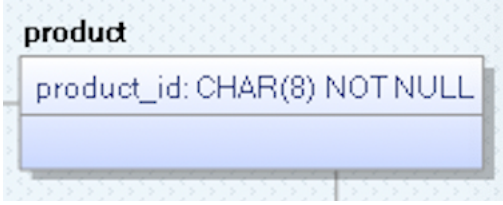
해당 프로젝트는 가상의 Package Delivery 회사의 데이터 베이스 시스템을 구축하는 것을 목표로 한다. 프로젝트1에서 구현한 E-R diagram 및 relational schema diagram을 기반으로 하여 physical diagram, query processing, relational database를 구현하고 유지한다.

2. entities and relationship

package delivery service를 제공하는 company의 entitiy 및 relationship을 소개한다. 이후 기존 프로젝트1과 비교하여 어느 부분에서 수정 및 decomposition이 일어났는지에 대해 설명하도록 한다.



2-1. Product

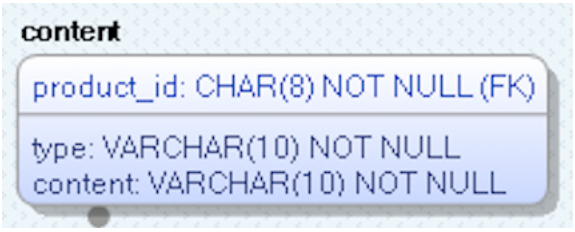


제품을 나타내는 스키마로, 제품에 대한 정보를 담고 있다. 하나의 package에 여러 제품이 담겨 배송될 수 있다는 점을 고려하여 제품을 하나의 스키마로 표현하였다. 이로 인해 package 스키마에 들어갈 제품을 나열할 필요없이, product 스키마를 통해 package에 어떤 제품이 들어갔는지, 총 몇개의 제품이 포함되어 있는지 나타낼 수 있다.

Product 스키마의 attribute는 Product_id로 제품의 고유 아이디를 의미한다. 특정한 경우를 제외하고는 제품이 무엇인지 알 필요가 없기에 제품에 구체적인 정보를 모두 포함하지

않고 제품의 id로만 제품을 구별하고자 한다. product_id의 domain은 8개의 char형이다. Product entity set의 primary key는 {product_id}로 구성되며, NULL을 허용하지 않는다. 해당 스키마의 functional dependency는 trivial한 {product_id -> product_id}가 유일하므로 BCNF의 조건을 만족한다.

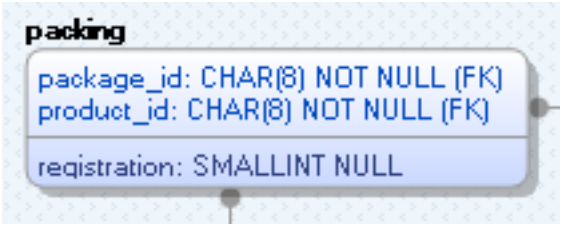
2-2. content



제품이 무엇인지를 나타내는 스키마로, 제품의 세부 정보를 담고 있다. 비록 package delivery 회사에서 어떤 물품을 배송하는지를 알 필요는 없지만, 위험물이나 해외로 배송되는 물품에 대해서는 택배 회사가 이에 대한 정보를 인지하고 있어야 한다.

content 스키마의 attribute는 총 3가지이다. product_id는 해당 물품의 고유 아이디를 의미한다. Type은 해당 product가 위험물 혹은 해외배송 물품인지를 나타낸다. Content는 위험물 혹은 해외배송 물품의 구체적인 정보를 나타낸다. 만약 위험물이라면, 해당 물품이 폭탄인지, 화학물품인지 등을 나타내게 된다. product_id는 product schema를 Referencing하는 foreign key이자 Primary key이며, domain은 8개의 char형이다. type, Content의 domain은 최대 10자리의 문자형이다. 모든 속성은 NULL을 허용하지 않는다. 해당 스키마의 functional dependency는 {product_id -> type, content}로 알파 파트가 슈퍼키이기 때문에 BCNF의 조건에 만족한다.

2-3. packing

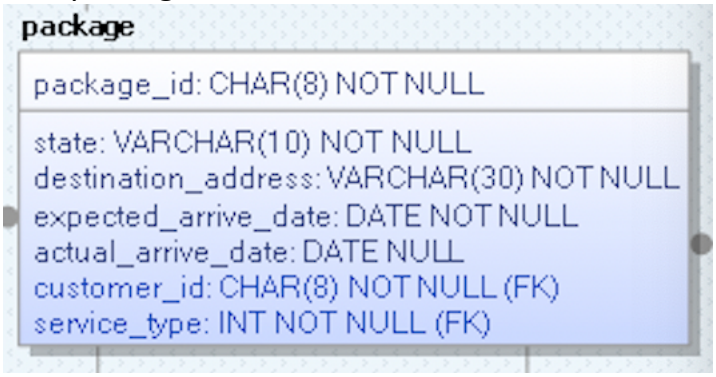


Product와 package를 연결하는 packing 스키마로, 하나의 패키지에 담긴 물건의 정보를 담고 있다.

packing 스키마의 속성은 총 3가지이다. package_id는 해당 패키지의 고유 아이디를, product_id는 해당 물품의 고유 아이디를 의미한다. registration은 해당 물품이 미리 신고가 필요한 물품일 경우 해당 물품이 신고를 했는지 여부를 나타낸다. package_id는 package 스키마를 referencing하는 foreign key이자 primary key이다. product_id 또한 product 스키마를 referencing하는 foreign key이자 primary key이다. 두 속성의 도메인은 8개의 문자형이며, NULL을 허용하지 않는다. registration의 도메인은 smallint형이며, 신고가 필요없는 물품일 경우 해당 값을 가지지 않아도 되므로, NULL을 허용한다. 해당 스키마의 functional

dependency는 {package_id, product_id -> registration}으로 알파 파트키 슈퍼키이기 때문에 BCNF를 만족한다.

2-4. package

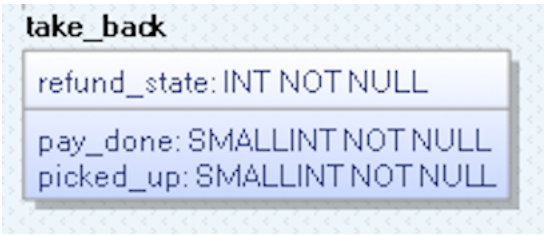


배송되는 택배 패키지에 대한 스키마로, 실제로 배송되는 패키지 단위를 나타내고 있다. package 스키마를 중심으로, 택배의 구성, 배송 추적, 고객에 대한 정보가 구성되기에 본 프로젝트의 핵심적인 스키마라 할 수 있다.

Package 스키마는 총 7개의 attribute로 구성된다. Package_id는 현재 패키지의 고유 번호를 나타내며, 실제 택배 서비스를 이용하면 나오는 송장번호와 유사한 개념이다. 8개의 문자로 구성되며 NULL값을 허용하지 않는다. State는 현재 패키지의 상태를 나타내기 위한 attribute로 창고에 있는 경우, 운송수단에 실려 배송중인 경우, 배송이 완료된 경우, 반품 절차를 진행 중인 경우를 나타낸다. 최대 10개의 문자로 구성되며 NULL값을 허용하지 않는다. Destination_address는 해당 패키지가 이동해야할 장소를 나타낸다. 최대 30개의 문자로 구성되며 NULL값을 허용하지 않는다. Expected_arrive_date는 예상 도착 일자로 패키지의 서비스 타입에 따른 배송 기간으로부터 일자를 구할 수 있다. date 데이터 형으로 구성되며 NULL값을 허용하지 않는다. Actual_arrive_date는 실제 도착 일자로 패키지가 배송이 완료되기 전까지는 Null값을 가지다가 배송완료 후 해당 값이 설정된다. date 데이터 형으로 구성되며 NULL값을 허용한다. 이후 예상 도착 일자와 실제 도착 일자와의 값을 비교하여 패키지가 제시간에 도착했는지 여부를 확인할 수 있다. customer_id는 해당 패키지를 배송하는 고객의 정보를 가지고 있다. 8개의 문자로 구성되며 NULL값을 허용하지 않는다. service_type은 해당 패키지의 서비스 타입을 의미하며, int로 구성되며 NULL값을 허용하지 않는다.

Package 스키마의 primary key는 {package_id}이다. 또한 foreign key는 customer를 referencing하는 customer_id와 service를 Referencing하는 service_type이 있다. 해당 스키마의 functional dependency는 {package_id -> state, destination_address, expected_arrive_date, actual_arrive_date, customer_id, service_type}으로 알파 파트가 슈퍼키이기 때문에 BCNF를 만족한다.

2-5. take_back

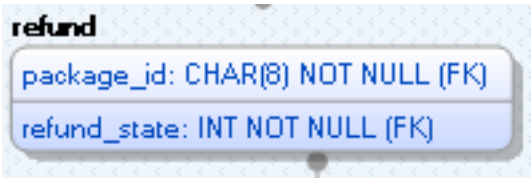


반품되는 패키지에 대한 스키마로, 반품되는 패키지에 대한 정보를 담고 있다. Package 중에서도 반품되는 패키지를 따로 구별하기 위해 스키마을 따로 구성하였다. 반품되는 상품은 본래 배송된 패키지와 별개의 package 스키마로 표현된다. 그 이유는 패키지 배송 사실을 남기기 위해, 패키지의 id와 다른 새로운 package_id를 부여해야 하고, 패키지 안에 구성되어 있는 모든 제품이 반품의 대상이 아닐 수도 있기 때문이다.

Take_ back 스키마은 총 3개의 attribute로 구성이 된다. refund_state는 반품 여부를 나타내며, 현재 반품과정 진행 상황이 어디까지 진행되었는지를 나타낸다. 해당 스키마의 primary key이며, 도메인은 int형이고 NULL값을 허용하지 않는다. Pay_done는 반품하는 패키지에 대한 반품비 결재 여부를 나타낸다. 도메인은 smallint이며 NULL값을 허용하지 않는다. Picked_up은 정상적인 반품여부를 나타낸다. 도메인은 smallint이며 NULL값을 허용하지 않는다. 패키지의 반품을 신청하였지만, 반품되는 물건이 없다거나, 잘못된 상품이 반품되는 경우를 대비하여 반품이 정상적으로 이루어졌는지를 나타낸다.

Take_back 스키마의 functional dependency는 {refund_state -> pay_done, picked_up }이다. 알파 파트가 슈퍼키이기 때문에 해당 스키마는 BCNF 조건을 만족한다.

2-6 refund

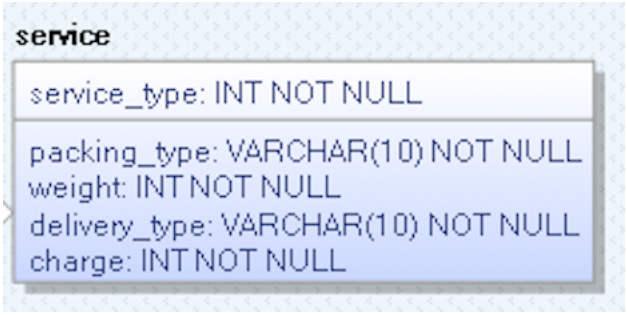


반품 상태를 나타내기 위한 스키마로, 반품되는 패키지와 반품 진행 상태를 나타내고 있다.

refund는 총 2개의 attribute로 구성이 된다. package_id는 반품 진행 중인 패키지를 가리키고 있으며, 해당 스키마의 primary key이자, package 스키마를 referencing하는 foreign key이다. 해당 속성의 도메인은 8개의 문자이며, NULL 값을 허용하지 않는다. refund_state는 현재 반품 진행 사항을 나타내는 속성으로, take_back 스키마를 referencing하는 foreign key이다. 해당 속성의 도메인은 int형이며, NULL값을 허용하지 않는다.

refund 스키마의 functional dependency는 {package_id -> refund_state}이다. 알파 파트가 슈퍼키이기 때문에 해당 스키마는 BCNF조건을 만족한다.

2-7 service

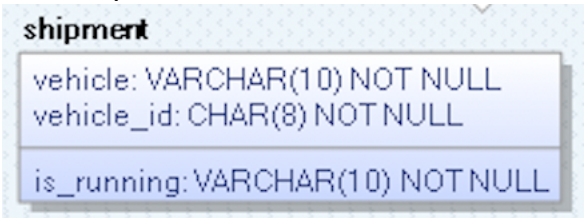


패키지에 적용되는 서비스의 종류를 나타내는 스키마로, 패킹의 타입(포장 종류), 패키지의 무게(무게 범위), 배송 시간, 가격에 따라 서비스 타입이 결정된다.

Service 스키마는 총 5개의 attribute로 구성된다. Packing_type은 패키지의 포장 방법을 의미한다. 도메인은 최대 10개의 문자이며, NULL값을 허용하지 않는다. Initialization에 언급되었듯이 플랫 봉투, 상자의 크기에 따라 구분할 수 있다. Weight는 패키지의 무게를 의미한다. 패키지의 정확한 무게를 일일이 저장한다면, 이에 따른 서비스 타입이 무한하기 때문에 일정 범위로 나누어서 구분하도록 한다. 도메인은 int이며, NULL값을 허용하지 않는다. Delivery_time은 패키지에 대한 배송 시간을 의미한다. 당일 배송, 1박 2일, 혹은 그 이상 등 배송에 걸리는 시간을 나타낸다. 도메인은 최대 10개의 문자이며, NULL값을 허용하지 않는다. charge는 해당 서비스의 가격을 의미하며, 도메인은 int이며, NULL값을 허용하지 않는다.

Service 스키마의 primary key는 {service_type}이다. 도메인은 int이며, NULL값을 허용하지 않는다. 해당 스키마의 functional dependency는 {service_type -> packing_type, weight, delivery_time, charge}로 알파 파트가 슈퍼키이기 때문에 BCNF를 만족한다.

2-8. shipment



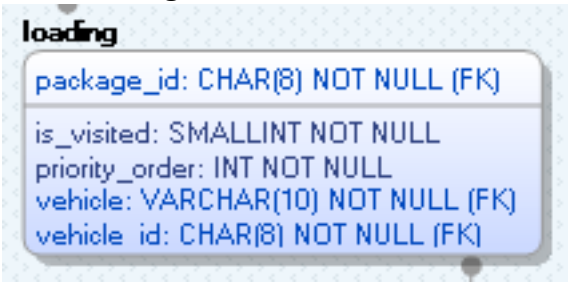
Shipment 스키마는 패키지를 운반하는 운행수단을 나타내는 스키마이다. 패키지를 운반하는 운행 수단의 종류(ex. 트럭, 선박, 항공기 등)와 그 운행수단의 고유 id와 현재 운행 여부를 나타낸다. Shipment 스키마는 운반하게 될 패키지와 관계를 이루게 되며 패키지를 목적지 혹은 허브와 같은 중간 집하 시설로 이동시키게 된다.

Shipment 스키마는 총 3개의 attribute로 구성된다. Vehicle은 운행 수단의 종류로 트럭, 선박, 항공기와 같이 분류되게 된다. 주로 패키지가 이동하는 시간과 거리에 따라 운반 수단이 결정될 것이다. 도메인은 최대 10개의 문자이며, NULL값을 허용하지 않는다. Vehicle_id는 이러한 운행 수단의 고유한 아이디로 같은 운반 수단 간을 구분하기 위해 사용하는 속성이다. 운행 수단과 운행 수단의 아이디를 통해 현재 운반 수단이 무슨 패키지를 운반하고 있으며, 이동 경로가 어떻게 되는지를 확인할 수 있다. 도메인은 8개의

문자이며, NULL값을 허용하지 않는다. Is_running은 현재 운반 수단의 운행 상태를 나타내는 속성이다. 현재 운행을 하며 패키지를 배송하고 있는지, 아직 패키지 배송을 시작하지 않았는지, 모든 패키지를 배송하고 운반을 마무리 했는지를 확인할 수 있다. 도메인은 최대 10개의 문자이며, NULL값을 허용하지 않는다.

Shipment 스키마의 primary key는 {vehicle, vehicle_id}이다. 운반 수단의 종류와 해당 운반 수단의 아이디로 모든 shipment entity를 구분할 수 있다. 아이디만으로는 운반수단이 다르지만 아이디가 같은 경우로 구별이 불가능할 경우를 대비해 두 속성을 모두 primary key로 설정하였다. 해당 스키마의 functional dependency는 {vehicle, vehicle_id -> is_running}으로 알파 파트가 슈퍼키에 해당하여 BCNF 조건을 만족한다.

2-9. loading

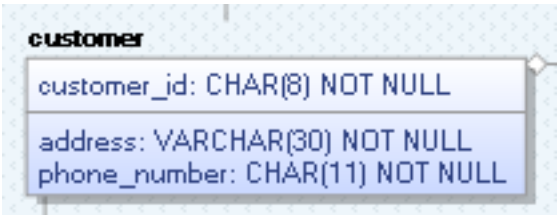


loading 스키마는 운송수단이 패키지를 운반하는 과정을 나타내고 있다. 한 운송 수단이 패키지를 적재하고 있다는 정보 뿐만 아니라 해당 패키지를 배송 완료하였는지 여부도 확인할 수 있다.

loading 스키마의 속성은 총 5개이다. package_id는 패키지의 고유 아이디로, 해당 스키마의 primary key이자, package 스키마를 referencing하는 foreign key이다. 도메인은 8개의 문자이며, NULL값을 허용하지 않는다. is_visited는 해당 패키지가 배송이 완료되었는지를 나타내는 속성으로 아직 차량에 적재되어 도착하지 않았다면 false를, 배송을 완료했다면 true 값을 가진다. 도메인은 smallint이며, NULL값을 허용하지 않는다. priority_order는 운송 순서를 나타내며, 본 가상 환경에서는 반드시 우선 순위 순으로 배송이 진행된다고 가정한다. 이를 통해 다음에 이동할 곳이 어디인지, 패키지가 배송되기 까지 얼마나 남았는지를 확인할 수 있다. 도메인은 int이며, NULL값을 허용하지 않는다. vehicle_id는 운송수단의 고유 아이디로, shipment 스키마를 referencing하는 foreign key이다. 도메인은 8개의 문자이며, NULL값을 허용하지 않는다. vehicle는 운송수단을 나타내며, shipment 스키마를 referencing하는 foreign key이다. 도메인은 최대 10개의 문자이며, NULL값을 허용하지 않는다.

loading 스키마의 functional dependency는 {package_id -> is_visited, priority_order, vehicle, vehicle_id}이며, 알파 파트가 슈퍼키에 해당하여 BCNF 조건을 만족한다.

2-10. customer

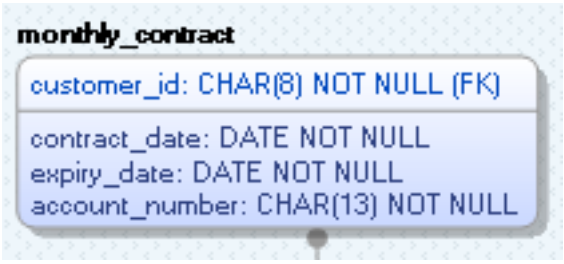


customer 스키마는 배송 서비스을 이용하는 고객의 정보를 나타낸다. 고객은 패키지 배송 서비스를 이용하는 사람으로 패키지를 보내는 사람을 가리킨다. 고객이 보낸 패키지의 내역, 월간 계약 여부, 과거 거래 정보를 확인할 수 있다.

Customer entity set의 attribute는 총 3가지 이다. Customer_id는 고객을 구분하기 위한 고유한 아이디이다. 고객을 구분하기 위해 아이디를 선택한 이유는 동명이인으로 인한 중복 문제를 배제시킴과 동시에 주민번호로 구분할 경우 발생할 수 있는 개인 정보 문제를 피하기 위함이다. 해당 속성은 primary key이며, 도메인은 8개의 문자이며, NULL값을 허용하지 않는다. Address는 고객의 주소로 만약 물품이 반품될 경우, 반품된 물건은 패키지를 배송한 고객의 주소로 배송하기 위함이다. 이로 인해 반품된 물건이 돌아갈 곳이 없는 이슈를 배제시킬 수 있다. 도메인은 최대 30개의 문자이며, NULL값을 허용하지 않는다. 마지막으로 phone_number는 고객에게 패키지의 상태를 알려주기 위한 정보이다. 패키지의 배송 시작, 배송 완료, 반품 접수 등 기본적인 정보와 더불어 패키지가 사고로 인해 문제가 생긴 경우에도 고객에게 공지할 수 있도록 하기 위해 해당 속성을 포함하였다. 도메인은 11개의 문자이며, NULL값을 허용하지 않는다.

customer 스키마의 functional dependency는 {customer_id -> address, phone_number}이며, 알파 파트가 슈퍼키에 해당하여 BCNF 조건을 만족한다.

2-11. monthly_contract

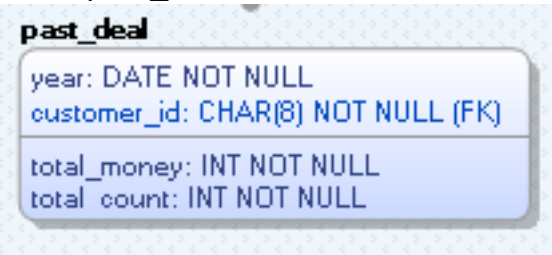


monthly_contract 스키마는 달마다 계약한 고객을 나타낸다. 고객이 패키지 서비스 회사와 계약을 했는지에 대한 여부와 계약을 했다면, 계약 시작일은 언제이며, 계약 만료일은 언제인지를 알 수 있다. 해당 스키마의 attribute는 총 4가지이다. 우선 customer_id는 고객을 구분하기 위한 속성으로 primary key이자, customer를 referencing하는 foreign key이다. 도메인은 8개의 문자이며, NULL값을 허용하지 않는다. Contract_date는 해당 고객의 계약 시작 일자이며, expiry_date는 해당 고객의 계약 만료 일자이다. 두 속성의 도메인은 date이며, NULL값을 허용하지 않는다. account_number는 계약 후 요금이 청구될 계좌번호를 나타내며, 도메인은 13개의 문자이며, NULL값을 허용하지 않는다.

monthly_contract 스키마의 functional dependency는 {customer_id -> contract_date, expiry_date,

account_number}이며, 알파 파트가 슈퍼키에 해당하여 BCNF 조건을 만족한다.

2-12. past_deal



past_deal 스키마는 1년간 해당 고객이 지출한 배송비와 배송한 패키지 개수를 나타낸다. 어느 고객이 1년간 가장 많은 돈을 지불하였는지, 가장 많은 패키지를 발송하였는지를 알 수 있도록 한다. Past_deal스키마의 attribute는 총 4가지이다. 우선 customer_id는 고객을 구분하기 위한 속성으로 primary key이자, customer를 referencing하는 foreign key이다. 도메인은 8개의 문자이며, NULL값을 허용하지 않는다. Year은 해당 집계를 기록한 년도를 의미한다. 한명의 고객이 다년간 패키지 배송 서비스를 이용할 경우, 년마다 소비 금액과 배송 개수가 다르므로 이를 여러해 동안의 정보를 저장할 수 있다. 따라서 해당 속성 역시 primary key이며, 도메인은 date이며, NULL값을 허용하지 않는다. Total_money는 해당 년도에 해당 고객이 패키지 배송에 소비한 금액을 의미한다. 도메인은 int이며, NULL값을 허용하지 않는다. Total_count는 해당 년도에 해당 고객이 배송한 패키지의 개수를 의미한다. 도메인은 int이며, NULL값을 허용하지 않는다. 이를 통해 해당 년도들을 비교하여 total_money와 total count가 가장 많은 고객을 도출해낼 수 있다.

past_deal 스키마의 functional dependency는 {year, customer_id -> total_money, total_count}이며, 알파 파트가 슈퍼키에 해당하여 BCNF 조건을 만족한다.

3. BCNF decomposition

앞서 소개한 스키마들은 모두 BCNF의 조건을 충족하고 있다. 이와 같은 logical diagram을 구성하기 위해 decomposition을 수행한 과정을 설명하고자 한다.

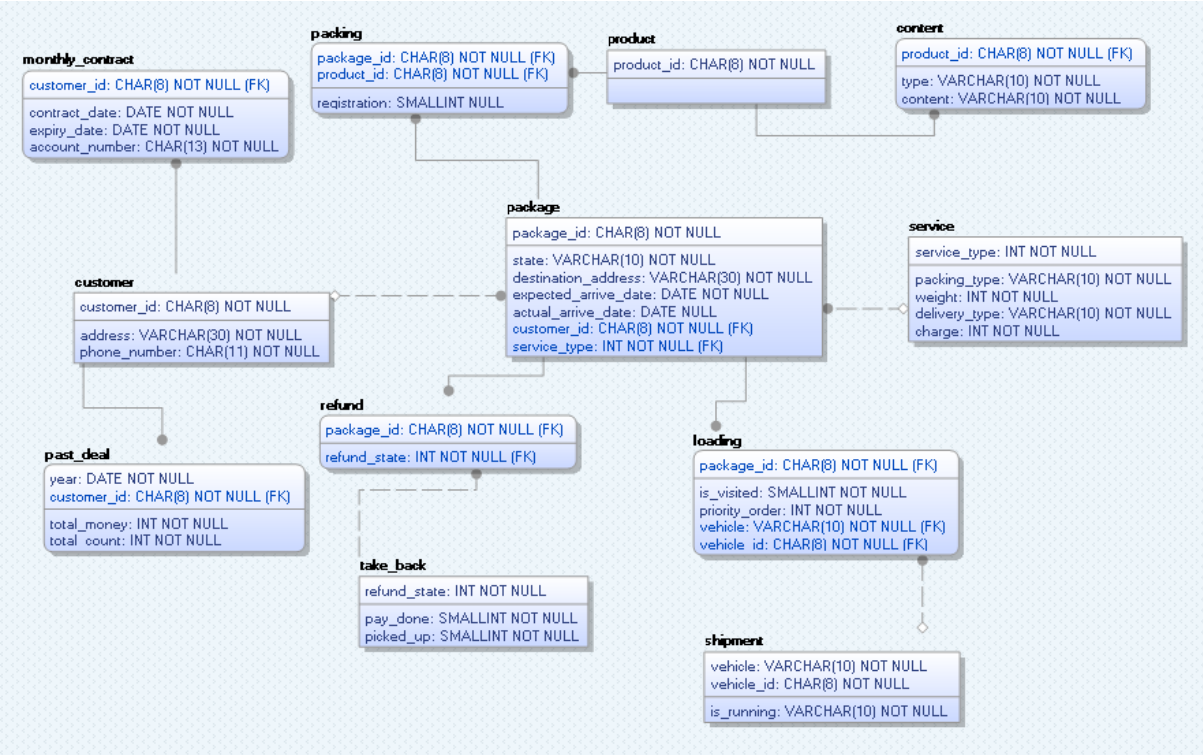
3-1. payment attribute in package entity set

기존 package에서 payment로 존재하던 속성이 중복 문제를 불러올 수 있다. 기존 프로젝트1에서 구성한 package에 customer_id,sevice_type 속성이 추가되어 프로젝트 2에서 package의 속성은 package_id, state, destination_address, expected_arrive_date, actual_arrive_date, service_type, payment, customer_id 총 8가지였다. 해당 스키마의 functional dependency는 {package_id -> state, destination, expected_arrive_date, actual_arrive_date, service_type, customer_id}, {service_type -> payment}가 존재하였다. 이는 서비스 종류를 통해 해당 비용을 계산할 수 있기 때문이다. 하지만 {service_type-> payment}는 알파 파트가 슈퍼키가 아니며, trivial하지도 않기 때문에 BCNF 조건을 위배하고 있다. 이러한 문제를 해결하기 위해, decomposition을 수행하였다. 한쪽은 {package_id, state,

destination_address, expected_arrive_date, actual_arrive_date, service_type, customer_id}로 분할 하였으며 해당 relation은 BCNF 조건을 만족한다. 또한 나머지 한쪽은 {serive_type, payment(=charge)}로 분할 되었고, 이 역시 BCNF 조건을 만족하여 lossless decomposition이 되었다. 이때, {service_type, payment}는 기존 service Relation에 추가되어 불필요한 분할을 막고자 하였다.

4. Physical schema diagram

물리적 스키마 다이어 그램은 다음과 같이 구성할 수 있다.



이러한 형태의 테이블이 실제 디스크에 저장되는 정보의 단위이다. 총 12개의 테이블로 구성되며, 이를 바탕으로 sql 테이블을 생성하고, 데이터를 삽입하며, 데이터를 관리한다. 앞서 1번 항목에서 서술한대로, 해당 속성의 이름과, 도메인, NULL값 허용 여부를 설정해 주었다. 뿐만 아니라 해당 테이블의 Primary Key, Foreign key 역시 설정해주었다. 이러한 physical diagram을 통해 sql의 create table 명령어를 작성할 수 있고, DDL.spl 파일에 이를 작성하였다. 또한 해당 테이블에 맞추어 insert into table values 명령어를 작성하여 datainsertion.sql 파일을 작성하였다. 마지막으로 생성된 테이블을 없애기 위한 drop 명령어를 작성하여 DROP.sql 파일을 작성하였다.

5. OBDC implementation within MySQL and code

Visual Studio 2022를 활용하여 C/C++에 SQL을 embedded하였다. C언어를 기반으로 코드를 작성하였으며, 구성은 크게, SQL 서버 연결 및 테이블과 데이터 초기화, 사용자 입력에 따른 쿼리문 수행, 테이블 drop와 프로그램 종료로 나뉜다.

5-1 SQL 서버 연결 및 테이블과 데이터 초기화

해당 프로그램을 실행하면, mysql_inti(), mysql_real_connect(), mysql_select_db()함수를 순서대로 호출하여 데이터베이스 시스템에 연결한다. 이후 사용자 정의 함수 init_mysql() 함수를 실행한다. 해당 함수는 데이터 베이스에서 쿼리문을 수행하기 위해 테이블을 정의하고 데이터를 삽입하는 명령어를 수행한다. 같은 파일 경로에 저장된 DDL.txt 파일을 읽어와 한줄씩 MySQL에 전달하여 테이블을 생성한다. 이후 datainsertion.txt파일을 읽어와 한줄씩 MySQL에 전달하여 테이블에 데이터를 삽입한다. 모든 과정이 정상적으로 수행되면 다음과 같은 상태가 된다.

```
Connection Succeed
complete to execute DDL.sql
complete to execute dateinsertion.sql
------- SELECT QUERY TYPES -------

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

5-2. 사용자 입력에 따른 쿼리문 수행

연결 및 초기화가 완료되면 무한 루프를 통해 사용자로부터 입력을 받는다. 사용자가 선택한 입력에 따른 쿼리문을 실행하도록 한다. handle_query() 함수를 호출하여 사용자 입력에 맞는 함수를 호출하도록 한다. 1번 쿼리문은 handle_type1()함수에서 처리하며 1번 쿼리에 대해서는 서브 쿼리가 존재하므로 사용자가 서브 쿼리를 선택하도록 한다. 서브 쿼리 1,2,3번에 대해 입력을 받아 이에 맞는 함수를 다시 한번 호출한다. 만약 0번일 들어오면, 무한 루프를 탈출하여 바깥 쿼리를 선택하는 쪽으로 돌아간다. 그렇지 않은 경우 서브 쿼리를 계속해서 선택할 수 있도록 한다. 그 결과 type1-1,1-2,1-3,2,3,4,5는 모두 같은 로직으로 수행되며, 이를 통합하여 설명하도록 한다. 우선 해당 타입의 쿼리가 저장되어 있는 파일에서 쿼리문을 읽어온다. 이때 사용자의 입력이 있는 경우에는 사용자의 입력부분에 개행 문자를 추가한다. 그 결과 첫번째 fgets에서 사용자의 입력이 들어갈 부분 직전까지만 쿼리문이 입력되고, 사용자로부터 값을 입력받아 문자열로 변경 후 strcat을 활용하여 쿼리문 뒤에 추가한다. 이후 나머지 문자열을 다시 fgets를 통해 파일로부터 읽어와 쿼리문 위에 추가한다. 그 결과 sql query가 buffer 문자 배열에 저장되게 된다. 이후 mysql_query(), mysql_store_result(), mysql_fetch_row(), mysql_free_result() 함수를 순서대로 실행하여 쿼리문을 전달하고 결과를 입력받아 터미널에 출력한다.

5-3. 테이블 drop 및 프로그램 종료

쿼리문을 수행하다가 사용자가 0을 입력할 경우 무한루프를 빠져나오게 된다. 이때 deinit_mysql(), mysql_close() 함수를 실행한 뒤, 프로그램을 종료한다. deinit_mysql()함수는

DROP.txt 파일을 읽어와 파일 내에 있는 drop문을 실행한다. 그 결과 sql서버에 올라와 있는 테이블이 모두 drop되어 초기 상태로 돌아간다. 이후 sql 서버와 연결을 끊고 프로그램을 종료하게 된다.

6. query execution

6-1. type1-1 query

해당 쿼리는 트럭 x가 파괴된 상황을 가정한다. 해당 트럭이 파괴되었을 때, 파괴된 트럭에 적재되어 있는 패키지를 보낸 손님을 찾고자 한다. 이를 위해 다음과 같이 쿼리를 작성하였다.

```
select distinct customer_id, package_id
from package natural join ( select package_id
                            from loading natural join shipment
                            where shipment.vehicle_id = '22222224'
                                  and loading.is_visited=false) as accident
```

우선 본 쿼리에서는 22222224번의 vehicle_id를 가진 트럭이 사고가 났다고 가정한다. 서브 쿼리를 통해 사고가 난 트럭에 적재되어 있는 패키지를 구한다. 우선 shipment와 loading의 natural join을 통해 어떤 트럭에 어떤 패키지가 적재되어 있는지를 구한다. 이후 사고 차량인 vehicle_id가 22222224번이고, 해당 트럭에 적재된 패키지들이 아직 배송되지 않은 경우(loading.is_visited=false)인 패키지를 구한다. 이렇게 구한 relation과 package를 natural join하여 해당 패키지를 발송한 고객이 누구인지를 매칭한다. 이후 해당 고객과 패키지 아이디를 선택한다. 해당 쿼리를 수행한 결과는 다음과 같다.

```
input query number : 1
------- Subtypes in TYPE I -------

        1. TYPE I-1
        2. TYPE I-2
        3. TYPE I-3
input subquery num : 1
----- TYPE I-1 ------

**find all customers who had a package on the truck at the time of the crash**
input truckNum : 22222224
customer_id     package_id
20230002        00000012
20230001        00000013
```

6-2. type1-2 query

해당 쿼리 역시 트럭 x가 파괴된 상황을 가정한다. 해당 트럭이 파괴되었을 때, 파괴된 트럭에 적재되어 있는 패키지를 받아야할 수신자를 찾고자 한다. 이를 위해 다음과 같이 쿼리를 작성하였다.

```
select distinct package.destination_address, package_id
from package natural join ( select package_id
                            from loading natural join shipment
                            where shipment.vehicle_id='22222224'
                                  and loading.is_visited=false) as accident
```

우선 본 쿼리에서는 22222224번의 vehicle_id를 가진 트럭이 사고가 났다고 가정한다. 서

브 쿼리를 통해 사고가 난 트럭에 적재되어 있는 패키지를 구한다. 우선 shipment와 loading의 natural join을 통해 어떤 트럭에 어떤 패키지가 적재되어 있는지를 구한다. 이후 사고 차량인 vehicle_id가 22222224번이고, 해당 트럭에 적재된 패키지들이 아직 배송되지 않은 경우(loading.is_visited=false)인 패키지를 구한다. 이렇게 구한 relation과 package를 natural join하여 해당 패키지를 받아야할 수신자가 누구인지를 매칭한다. 이후 해당 수신자의 주소와 패키지 아이디를 선택한다. 이때 주소만을 출력하는 이유는 택배회사 입장에서 수신자의 주소 이외에 더 많은 정보를 알아야할 이유가 없다고 판단하였고, 그렇기 때문에 수신자의 정보인 주소를 선택한다. 해당 쿼리를 수행한 결과는 다음과 같다.

```
input subquery num : 2
----- TYPE I-2 ------

**find all recipients who had a package on that truck at the time of the crash**
input truckNum : 22222224
destination_address     package_id
4 street 11 home        00000012
4 street 12 home        00000013
```

6-3 type1-3 query

해당 쿼리 역시 트럭 x가 파괴된 상황을 가정한다. 해당 트럭이 파괴되었을 때, 파괴된 트럭에서 마지막으로 성공한 배송에 대한 정보를 찾고자 한다. 이를 위해 다음과 같이 쿼리를 작성하였다.

```
select distinct package.customer_id, package.package_id
from package natural join ( select *
                            from loading natural join shipment
                            where shipment.vehicle_id='22222224'
                                  and loading.is_visited=true
                            order by priority_order desc limit 1) as accident
```

우선 본 쿼리에서는 22222224번의 vehicle_id를 가진 트럭이 사고가 났다고 가정한다. 서브 쿼리를 통해 사고가 난 트럭에 적재되어 있는 패키지를 구한다. 우선 shipment와 loading의 natural join을 통해 어떤 트럭에 어떤 패키지가 적재되어 있는지를 구한다. 이후 사고 차량인 vehicle_id가 22222224번이고, 해당 트럭에 적재된 패키지들이 이미 배송된 경우(loading.is_visited=true)인 패키지를 구한다. 이후 운송 순서를 저장하고 있는 priority_order를 내림차순으로 정렬하여 첫번째 패키지를 구한다. 이렇게 구한 relation과 package를 natural join하여 해당 배송에 대한 정보를 선택한다. 해당 쿼리를 수행한 결과는 다음과 같다.

```
input subquery num : 3
----- TYPE I-3 ------

**find the last successful delivery by that truck prior to the crash**
input truckNum : 22222224
customer_id     package_id
20230001        00000011
```

6-4 type2 query

해당 쿼리는 입력으로 주어진 년도의 이전 년도를 구하여 해당 년도에 가장 많은 패키지를 발송한 고객을 찾고자 한다. 이를 위해 다음과 같이 쿼리를 작성하였다.

```
select customer_id, past.total_count, past_year
from (select *
      from past_deal
      where past_year='2023'
      order by total_count desc limit 1) as past
```

우선 본 쿼리는 입력으로 2024년이 들어와 그 이전해인 2023년의 정보를 구한다고 가정한다. 서브 쿼리를 통해 과거 거래 정보를 저장하고 있는 past_deal relation에서 해당 년도가 2023년인 정보를 구한다. 이를 total_count를 기준으로 내림차순으로 정렬하여 가장 큰 데이터만을 추출하여 해당 년도에 가장 많은 패키지를 보낸 고객의 정보를 선택한다. 이후 해당 고객의 아이디와, 배송개수, 년도를 선택한다. 해당 쿼리를 수행한 결과는 다음과 같다.

```
input subquery num : 0
------- SELECT QUERY TYPES -------

        1. TYPE 1
        2. TYPE 2
        3. TYPE 3
        4. TYPE 4
        5. TYPE 5
        0. QUIT
input query number : 2
----- TYPE II ------

**find the customer who has shipped the most packages in the past year**
input year : 2024
customer_id     total_count     past_year
20230003        38              2023
```

6-5 type3 query

해당 쿼리는 입력으로 주어진 년도의 이전 년도를 구하여 해당 년도에 가장 많은 돈을 소비한 고객을 찾고자 한다. 이를 위해 다음과 같이 쿼리를 작성하였다.

```
select customer_id, past.total_money, past_year
from (select *
      from past_deal
      where past_year='2023'
      order by total_money desc limit 1) as past
```

우선 본 쿼리는 입력으로 2024년이 들어와 그 이전해인 2023년의 정보를 구한다고 가정한다. 서브 쿼리를 통해 과거 거래 정보를 저장하고 있는 past_deal relation에서 해당 년도가 2023년인 정보를 구한다. 이를 total_money를 기준으로 내림차순으로 정렬하여 가장 큰 데이터만을 추출하여 해당 년도에 가장 많은 돈을 소비한 고객의 정보를 선택한다. 이후 해당 고객의 아이디와, 소비금액, 년도를 선택한다. 해당 쿼리를 수행한 결과는 다음과 같다.

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

        1. TYPE 1
        2. TYPE 2
        3. TYPE 3
        4. TYPE 4
        5. TYPE 5
        0. QUIT
input query number : 3
----- TYPE III ------

**find the customer who has spent the most money on shipping in the past year**
input year : 2024
customer_id     total_money     past_year
20230004        80000           2023
```

6-6 type4 query

해당 쿼리는 지금까지 발송하기로 한 시간을 어겨서 배달된 패키지를 구하고자 한다. 이를 위해 다음과 같은 쿼리를 작성하였다.

```
select customer_id, package_id, expected_arrive_date, actual_arrive_date
from package
where state='complete' and expected_arrive_date != actual_arrive_date
```

패키지 relation에 저장된 배송 완료된(state='complete') 패키지에 대해 예상 도착일과 실제 도착일을 비교한다. 이때, 예상 도착일보다 빨리 도착하는 경우는 없다고 가정하며, 두 날짜의 차이가 있다면 늦게 도착한 것이 되도록 한다. 해당 조건을 만족하는 패키지와 고객에 대한 정보 예상도착일과 실제 도착일을 선택한다. 해당 쿼리를 수행한 결과는 다음과 같다.

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

        1. TYPE 1
        2. TYPE 2
        3. TYPE 3
        4. TYPE 4
        5. TYPE 5
        0. QUIT
input query number : 4
----- TYPE IV ------

**find the packages that were not delivered whitin the promised time**
customer_id     package_id      expected_arrived_date actual_arrive_date
20230003        00000003        2023-06-07              2023-06-08
20230004        00000004        2023-06-07              2023-06-08
20230005        00000005        2023-06-07              2023-06-09
20230006        00000006        2023-06-07              2023-06-09
```

6-7 type5 query

해당 쿼리는 사용자가 입력한 년도와 달을 입력하여 이 전의 달의 고객에 대한 영수증을 출력하고자 한다. 이를 위해 다음과 같이 쿼리를 작성하였다.

```
select customer.customer_id, customer.address, A.sumcharge
from customer natural join (select package.customer_id,
                                   sum(service.charge) as sumcharge
```

```
                        from package natural join service
                        where package.actual_arrive_date>='2023-06-01'
                              and package.actual_arrive_date<'2023-07-01'
                        group by package.customer_id) as A
```

서브 쿼리를 통해 해당 기간을 조건으로 하여 고객을 기준으로 그룹을 구성한다. 해당 고객이 패키지 배송을 위해 지불한 돈을 sum을 통해 구한다. 이를 customer relation과 natural join하여 해당 고객 정보와 고객이 지불한 돈을 선택하여 하나의 영수증을 구성한다. 해당 쿼리를 수행한 결과는 다음과 같다.

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

        1. TYPE 1
        2. TYPE 2
        3. TYPE 3
        4. TYPE 4
        5. TYPE 5
        0. QUIT
input query number : 5
----- TYPE V ------

**generate the bill for each customer for the past month**
input year : 2023
input_month : 7
customer_id     customer_address        sum of charge
20230005        1 street 5 home         5000
20230006        2 street 1 home         5000
20230004        1 street 4 home         4000
20230003        1 street 3 home         3000
20230002        1 street 2 home         2000
20230001        1 street 1 home         1000
```