|  |
| --- |
| **데이터베이스(1조)**  **데이터베이스 설계 과제** |

****

|  |  |
| --- | --- |
| **과목** | **데이터베이스** |
| **담당 교수님** | **허훈식 교수님** |
| **제출 과제** | **데이터베이스 설계 과제** |
| **제출 마감일** | **2019.12.03** |
| **제출자** | **김영민 2016156007**  **이영채 2016156022**  **최성훈 2016156039**  **황규빈 2016156047** |

****

**목차**

1. **업무 분석**

1-1 설계목적 및 범위설정  **- - - - - - - - - - - - - -**  2p 1-2 요구사항분석  **- - - - - - - - - - - - - -**  3p

1. **개념적 설계**

2-1 엔티티와 애트리뷰트  **- - - - - - - - - - - - - - -**  4p

2-2 관계설정하기 **- - - - - - - - - - - - - - -**  5p

2-3 ER다이어그램 **- - - - - - - - - - - - - - -** 10p

1. **논리적 설계**

3-1 ER-관계 사상 알고리즘 **- - - - - - - - - - - - -**  11p

3-2 정규화 **- - - - - - - - - - - - - - -**  14p

1. **물리적 설계 - - - - - - - - - - - - - - -**  15p
2. **기능수행 질의문**

5-1 ER-관계 사상 알고리즘 **- - - - - - - - - - - - -**  20p

5-2 서비스 기능 질의문 **- - - - - - - - - - - - - - -**  24p

5-3 뷰 생성 **- - - - - - - - - - - - - - -**  34p

1. **기타 - - - - - - - - - - - - - - -**  36p

**1. 요구 사항 분석**

**1-1 설계 목적 및 범위 설정**

* 시스템 설계 목적

- 우체국의 다양한 서비스 중, 택배, 우편, 예금의 서비스를 기준으로 보다 실제 환경과 유사하게 만들고자 하였습니다.

그리고 실제 우체국 비즈니스에서 주요하게 사용하는 기능,

대표적인 예로 택배의 배송 조회, 우편의 수신 여부, 예금의 계좌 조회 등을 인터넷을 통해 사용할 수 있도록 하는 것을 논리적으로 선택하고, 구현하고자 하였습니다.

* 설계 범위 설정

- 우체국을 크게 택배, 우편, 예금으로 나눠 고객이 사용할 수 있도록 하였고, 택배와 우편을 배송하는 기사와 우체국에서 업무를 담당하는 직원 그리고 예금을 예금상품과 계좌정보로 설계 범위를 설정하였다.

**1. 요구 사항 분석**

**1-2 요구사항 분석**

* 택배는 등기번호를 기본키로 하여, 받는 분의 주소와 전화번호, 보내는 분의 주소와 전화번호, 현재위치, 배송 진행 상태, 물품 정보, 접수시간을 저장한다.
* 택배는 1개의 택배 종류를 가져야 한다.
* 우편은 등기번호를 기본키로 하여, 받는 분의 주소와 전화번호, 보내는 분의 주소와 전화번호, 수신여부, 접수시간을 저장한다.
* 우편은 1개의 우편 종류를 가져야 한다.
* 택배와 우편은 현재위치와 수신여부를 조회를 할 수 있어야 하고, 조회는 고객의 ID와 등기번호를 사용한다.
* 기사는 자신이 배송해야 하는 우편과 택배를 조회할 수 있다.
* 예금은 고객이 여러 개의 계좌 정보를 가질 수 있고, 계좌정보는 계좌번호를 기본키로 하여, 비밀번호, 잔액, 개설일을 저장한다.
* 계좌정보는 어떠한 예금상품에 속하는지 알아야 하는데, 예금상품이라는 테이블에 예금상품번호를 기본키로 하여, 예금종류, 예금이름, 가입기간, 적용이율을 저장하고 계좌정보는 예금상품과 관계를 맺는다.
* 직원은 본인이 소속된 부서를 한가지만 가질 수 있다.
* 고객은 본인의 우편번호가 입력되면 본인의 우편번호를 알 수 있다.

**2. 개념적 설계**

**2-1 엔티티와 애트리뷰트 결정하기**

우편

- 등기번호

- 받는 분 이름

- 받는 분 주소

- 받는 분 번호

- 보내는 분 이름

- 보내는 분 주소

- 보내는 분 번호

- 수신 여부

- 접수 시간

택배

- 등기번호

- 받는 분 이름

- 받는 분 주소

- 받는 분 번호

- 보내는 분 이름

- 보내는 분 주소

- 보내는 분 번호

- 현재 위치

- 배송 진행 상태

- 물품 정보

- 접수 시간

고객

- 고객 ID

- 이름

- 연락처

- 비밀번호

택배 종류

- 택배 종류 번호

- 택배 종류

우편 종류

- 우편 종류 번호

- 우편 종류

우편 번호

- 주소

- 우편 번호

기사

- 기사 번호

- 이름

- 연락처

- 주소

- 주민 번호

직원

- 직원 번호

- 이름

- 주소

- 연락처

- 주민 번호

계좌 정보

- 계좌 번호

- 비밀 번호

- 잔액

- 개설

부서

- 부서 번호

- 부서 이름

예금 종류

- 예금 종류 번호

- 예금 이름

예금 상품

- 예금 상품 번호

- 이율

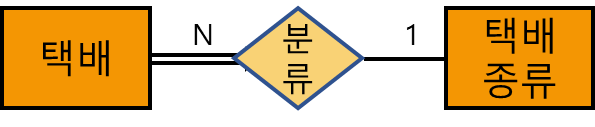
- 예금 상품 이름

- 기간

**2. 개념적 설계**

**2-2 관계 설정하기**

* **택배 – 택배종류**



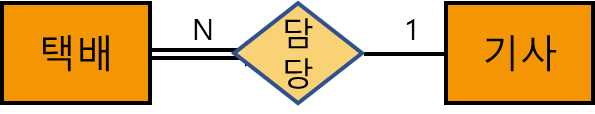
우편 종류

기사

직원

택배는 택배종류와 분류라는 관계를 가지며, 하나의 택배는 하나의 택배종류를 무조건 가져야 하고, 택배종류는 여러가지의 택배를 포함할 수 있다.

* **택배 – 기사**



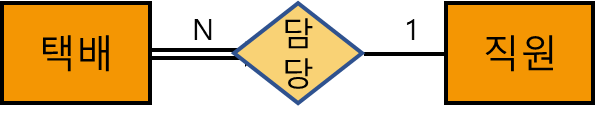
택배는 기사와 담당이라는 관계를 가지며, 하나의 택배는 무조건 한 명의 기사에게 속해야 하며, 기사는 여러가지의 택배를 배송할 수 있다.

* **택배 – 고객**



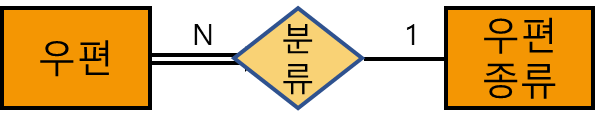
택배는 고객과 접수라는 관계를 가지며, 하나의 택배는 무조건 한 명의 고객에 의해서 발송되며, 고객은 여러가지의 택배를 발송 할 수 있다.

* **택배 – 직원**



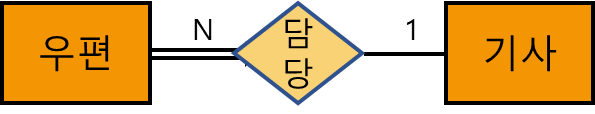
택배는 직원과 담당이라는 관계를 가지며, 하나의 택배는 무조건 한 명의 직원에 의해서 담당되고, 직원은 여러가지 택배를 담당 할 수 있다.

* **우편 – 우편 종류**



우편은 우편종류와 분류라는 관계를 가지며, 하나의 우편은 하나의 우편종류를 무조건 가져야 하고, 우편종류는 여러가지의 우편을 포함할 수 있다.

* **우편 – 기사**



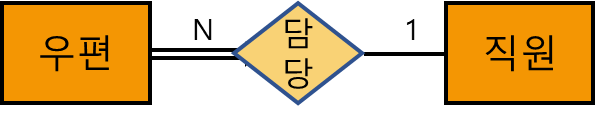
우편은 기사와 담당이라는 관계를 가지며, 하나의 우편은 무조건 한 명의 기사에게 속해야 하며, 기사는 여러가지의 우편을 배송할 수 있다.

* **우편 – 고객**

****

우편은 고객과 접수라는 관계를 가지며, 하나의 우편은 무조건 한 명의 고객에 의해서 발송되며, 고객은 여러가지의 우편을 발송 할 수 있다.

* **우편 – 직원**



우편은 직원과 담당이라는 관계를 가지며, 하나의 우편은 무조건 한 명의 직원에 의해서 담당되고, 직원은 여러가지 우편을 담당할 수 있다.

* **계좌 정보 – 고객**

****

계좌 정보는 고객과 소유라는 관계를 가지며, 하나의 계좌 정보는 무조건 한 명의 고객에 의해 소유되며, 고객은 여러가지의 계좌정보를 가질 수 있다.

* **계좌 정보 – 예금 상품**

****

계좌 정보는 예금상품과 상품이라는 관계를 가지며, 하나의 계좌 정보는 무조건 하나의 예금 상품을 갖게 되고, 예금 상품은 여러가지의 계좌 정보를 가질 수 있다.

* **계좌 정보 – 직원**

****

계좌 정보는 직원과 개설이라는 관계를 가지며, 하나의 계좌 정보는 무조건 한 명의 직원에 의해 개설되며, 직원은 여러 개의 계좌를 개설할 수 있다.

* **고객 – 우편번호**



고객은 우편번호와 주소라는 관계를 가지며, 한 명의 고객은 무조건 하나의 우편번호를 주소와의 관계에 의해 갖게 된다. 우편 번호는 여러 명의 고객과 관계를 가질 수 있다.

* **예금 상품 – 예금 상품 종류**

****

예금 상품은 예금 상품 종류와 분류라는 관계를 가지며, 하나의 예금 상품은 무조건 하나의 예금 상품 종류를 가져야 하며, 예금 상품 종류는 여러가지의 예금 상품을 포함할 수 있다.

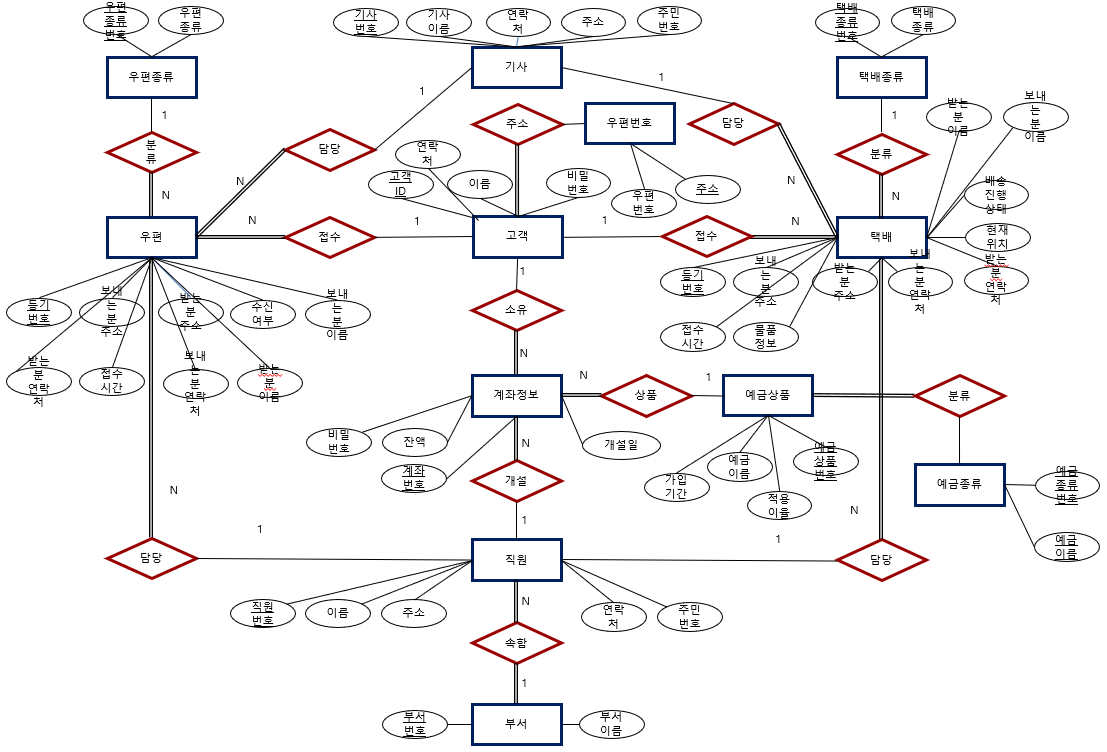
* **직원 – 부서**

****

직원은 부서와 속함이라는 관계를 가지며, 한 명의 직원은 무조건 하나의 부서를 가져야 하고, 부서는 1명 이상의 직원들을 포함하고 있어야한다.

**2. 개념적 설계**

**2-3 ER다이어그램**



**3. 논리적 설계**

**3-1 ER-관계 사상 알고리즘**

- **단계1** : 정규 엔티티 타입과 단일 값 애트리뷰트

ER 스키마의 각 정규 엔티티 타입에 대해 하나의 릴레이션을 생성했다.

**PARCEL**(REGINO, TO\_NAME, TO\_ADDR, TO\_TEL, FROM\_NAME, FROM\_ADDR, FROM\_TEL, CURRENT\_LOC,DELIVERY\_STATE, PRODUCT\_DATA, RECEPTION\_TIME)

**PARCEL\_TYPE**(PTYPENO, PARCELTYPE)

**MAIL**(REGINO, TO\_NAME, TO\_ADDR, TO\_TEL, FROM\_NAME, FROM\_ADDR, FROM\_TEL, REG\_TIME, REC\_STATUS)

**MAIL\_TYPE**(MTYPENO, MAILTYPE)

**POST\_CODE**(ADDR, POST\_CODE)

**ACCOUNT\_DATA**(ACCNO, PASSWD, BALANCE, OPEN\_DATE)

**CUSTOMER**(CUSTID, NAME, FK\_ADDR, TEL, PASSWORD)

**DELIVERY\_MAN**(DELNO, DELNAME, RESIDENT\_NO, ADDR, TEL)

**DEPARTMENT**(DEPTNO, DEPTNAME)

**DEPOSIT\_PRODUCT**(DEPONO, DEPO\_RATE, DEPONAME, PERIOD)

**DEPOSIT\_TYPE**(DEPNO, DEPOTYPE)

**EMPLOYEE**(EMPNO, NAME, ADDR, TEL, RESIDENT\_NO)

- **단계2** : 약한 엔티티 타입과 단일 값 애트리뷰트

약한 엔티티 타입이 존재하지 않기 때문에 만족한다.

- **단계3** : 2진 1 : 1 관계타입

1 : 1 관계가 존재하지 않기 때문에 만족한다.

- **단계4** : 2진 1 : N 관계타입

1 : N 관계에서 외래키를 릴레이션에 추가해주었다.

**PARCEL**(REGINO, TO\_NAME, TO\_ADDR, TO\_TEL, FROM\_NAME, FROM\_ADDR, FROM\_TEL, CURRENT\_LOC,DELIVERY\_STATE, PRODUCT\_DATA, RECEPTION\_TIME, FK\_PTYPENO, FK\_DELNO, FK\_EMPNO, FK\_CUSTID)

**PARCEL\_TYPE**(PTYPENO, PARCELTYPE)

**MAIL**(REGINO, TO\_NAME, TO\_ADDR, TO\_TEL, FROM\_NAME, FROM\_ADDR, FROM\_TEL, REG\_TIME, REC\_STATUS, FK\_MTYPENO, FK\_DELNO, FK\_EMPNO, FK\_CUSTID)

**MAIL\_TYPE**(MTYPENO, MAILTYPE)

**POST\_CODE**(ADDR, POST\_CODE)

**ACCOUNT\_DATA**(ACCNO, PASSWD, BALANCE, OPEN\_DATE, FK\_CUSTID, FK\_DEPONO, FK\_EMPNO)

**CUSTOMER**(CUSTID, NAME, FK\_ADDR, TEL, PASSWORD)

**DELIVERY\_MAN**(DELNO, DELNAME, RESIDENT\_NO, ADDR, TEL)

**DEPARTMENT**(DEPTNO, DEPTNAME)

**DEPOSIT\_PRODUCT**(DEPONO, DEPO\_RATE, DEPONAME, PERIOD, FK\_DEPNO)

**DEPOSIT\_TYPE**(DEPNO, DEPOTYPE)

**EMPLOYEE**(EMPNO, NAME, ADDR, TEL, RESIDENT\_NO, FK\_DEPTNO)

- **단계5** : 2진 M : N 관계타입

M : N 관계가 존재하지 않기 때문에 만족한다.

- **단계6** : 3진 이상의 관계 타입

3진 이상의 관계가 존재하지 않기 때문에 만족한다.

- **단계7** : 다치 애트리뷰트

다치 애트리뷰트가 존재하지 않기 때문에 만족한다.

**3-2 정규화**

**제 1 정규형**에서는 모든 릴레이션의 모든 애트리뷰트가 원자 값만 갖습니다. 곧 복합, 다치 애트리뷰트가 존재하지 않아 이미 1정규형을 만족합니다.

**제 2 정규형**에서 직원과 기사의 릴레이션에 직원번호, 기사번호가 기본 키인 상황에서 주민번호도 기본 키가 될 수 있다고 판단하여, 새로운 릴레이션으로 분리를 해야 하는지에 대해 고민을 했습니다. 주민번호도 고유하게 식별하게 될 수 있지 않는지 생각을 해보았을 때, 주민번호가 동일한 인스턴스가 생길 수 있다고 생각해서 같은 릴레이션 안에 넣기로 했습니다. 예를 들어 퇴사했다가 재입사의 경우에도 직원번호는 달라졌지만 주민번호는 같기 때문에 고유하게 식별한다고 볼 수 있기 때문입니다. 직원, 기사 릴레이션을 뺀 나머지 릴레이션에서는 기본 키가 하나의 애트리뷰트로만 이루어져 있어서 2정규형도 만족합니다.

**제 3 정규화** 과정에서 기본 키가 아닌 모든 애트리뷰트들이 기본키에만 의존하기 때문에 제 3정규형을 만족합니다.

**BCNF**에서 제 3정규형을 만족하고, 모든 결정자가 후보 키이므로 만족합니다.

**기타**단계에서 예금 상품과 예금 종류 간의 관계에서 갱신이상, 즉 하나의 특정 상품 종류의 마지막 레코드를 삭제할 경우, 해당 상품 종류에 해당하는 레코드 정보가 모두 삭제되는 상황을 발견했습니다.

이를 해결하고자, ER 다이어그램으로 돌아가서 엔티티를 예금 상품, 예금 종류로 분리하였습니다

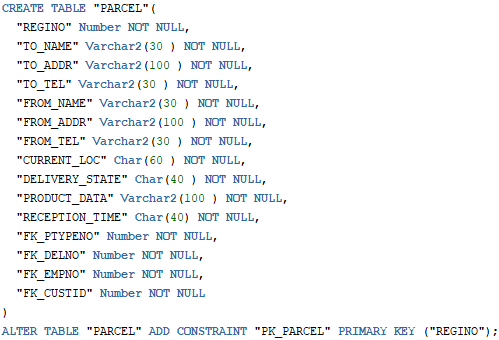
**4. 물리적 설계**

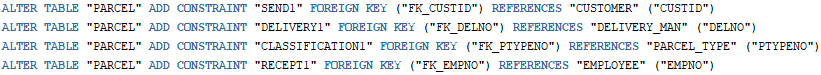
* **PARCEL(택배)**

- 기본키 : REGINO.

- 외래키 : CUSTOMER(CUSTID), DELIVERY\_MAN(DELNO),

PARCEL\_TYPE(PTYPENO), EMPLOYEE(EMPNO)

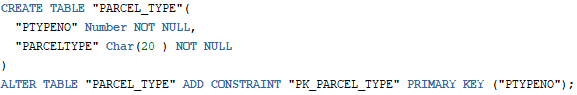






* **PARCEL\_TYPE(택배 종류)**

- 기본키 : PTYPENO



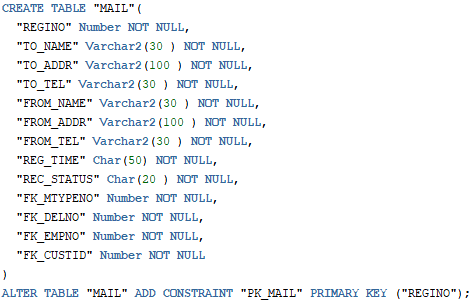


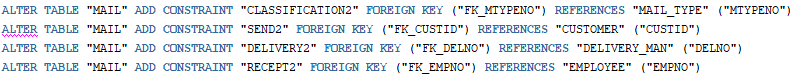
* **MAIL(우편)**

- 기본키 : REGINO

- 외래키 : CUSTOMER(CUSTID), DELIVERY\_MAN(DELNO),

PARCEL\_TYPE(PTYPENO), EMPLOYEE(EMPNO)



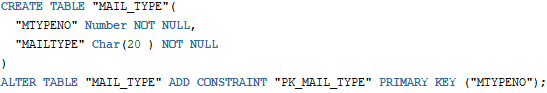






* **MAIL\_TYPE(우편 종류)**

- 기본키 : MTYPENO

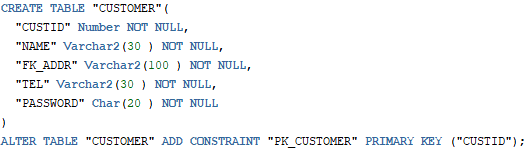




* **CUSTOMER(고객)**

- 기본키 : CUSTID

- 외래키 : POST\_CODE(ADDR)

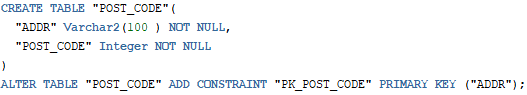






* **POST\_CODE(우편 번호)**

- 기본키 : ADDR



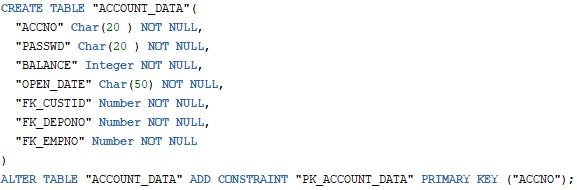


* **ACCOUNT\_DATA(계좌정보)**

- 기본키 : ACCNO

- 외래키 : CUSTOMER(CUSTID), EMPLOYEE(EMPNO),

DEPOSIT\_PRODUCT(DEPONO)

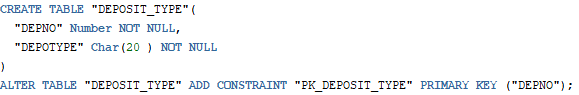






* **DEPOSIT\_TYPE(예금 상품 종류)**

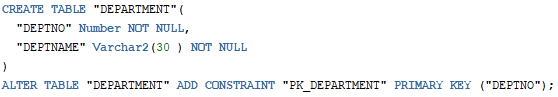
- 기본키 : DEPNO





* **DEPARTMENT(부서)**

- 기본키 : DEPTNO

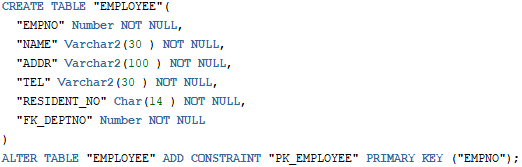




* **EMPLOYEE(직원)**

- 기본키 : EMPNO

- 외래키 : DEPARTMENT(DEPTNO)

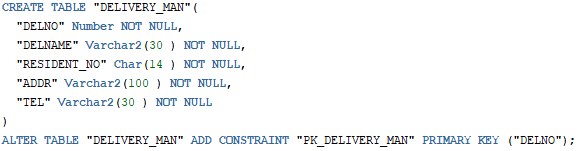






* **DELIVERY\_MAN(기사)**

- 기본키 : DELNO



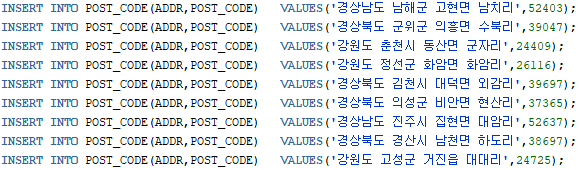


**5. 기능 수행 질의문**

* **테이블에 데이터 삽입**
  + **고객 정보 삽입**

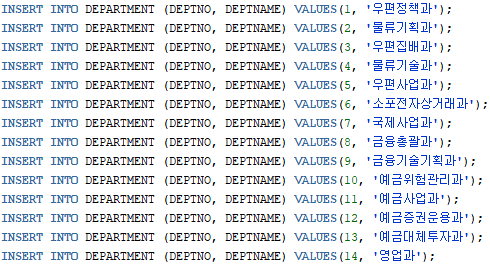


* + **우편 번호 정보 삽입**



\*우편 번호의 데이터는 우체국에서 제공하는 데이터를 가져왔습니다.

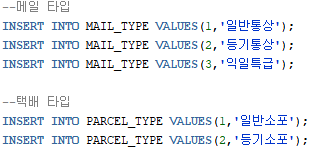
* + **부서 정보 삽입**



* + **직원 정보 삽입**



* + **택배, 우편 종류 정보 삽입**



* + **예금 종류와 예금 상품 정보 삽입**



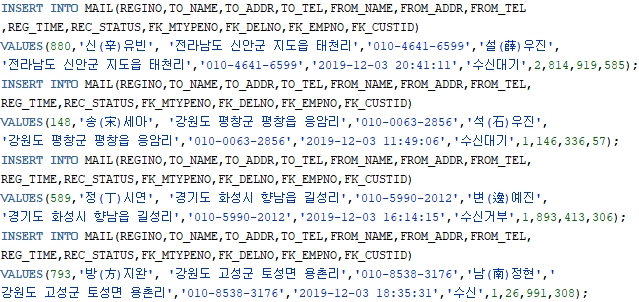
* + **기사 정보 삽입**



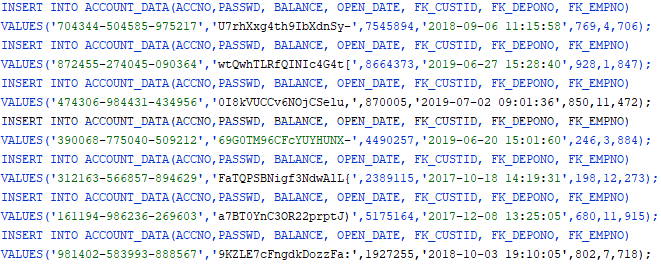
* + **택배 정보 삽입**



* + **우편 정보 삽입**

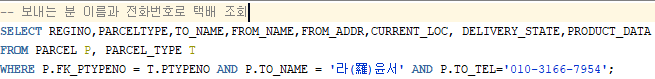


* + **계좌 정보 삽입**



* **서비스 기능 질의문**
* **보내는 분 이름과 전화번호로 택배 조회**

질의

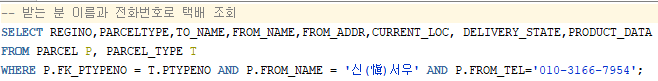


결과



* **받는 분 이름과 전화번호로 택배 조회**

질의



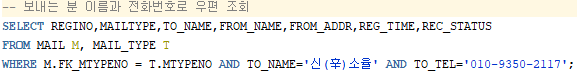
결과





* **보내는 분 이름과 전화번호로 우편 조회**

질의

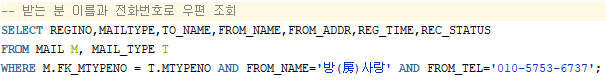


결과



* **받는 분 이름과 전화번호로 우편 조회**

질의



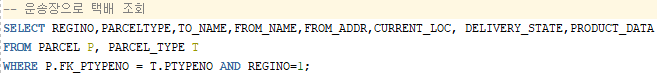
결과

****



* **운송장으로 택배 조회**

질의

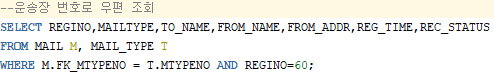


결과



* **운송장으로 우편 조회**

질의



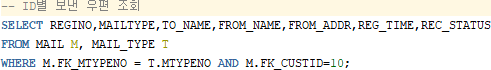
결과

****

****

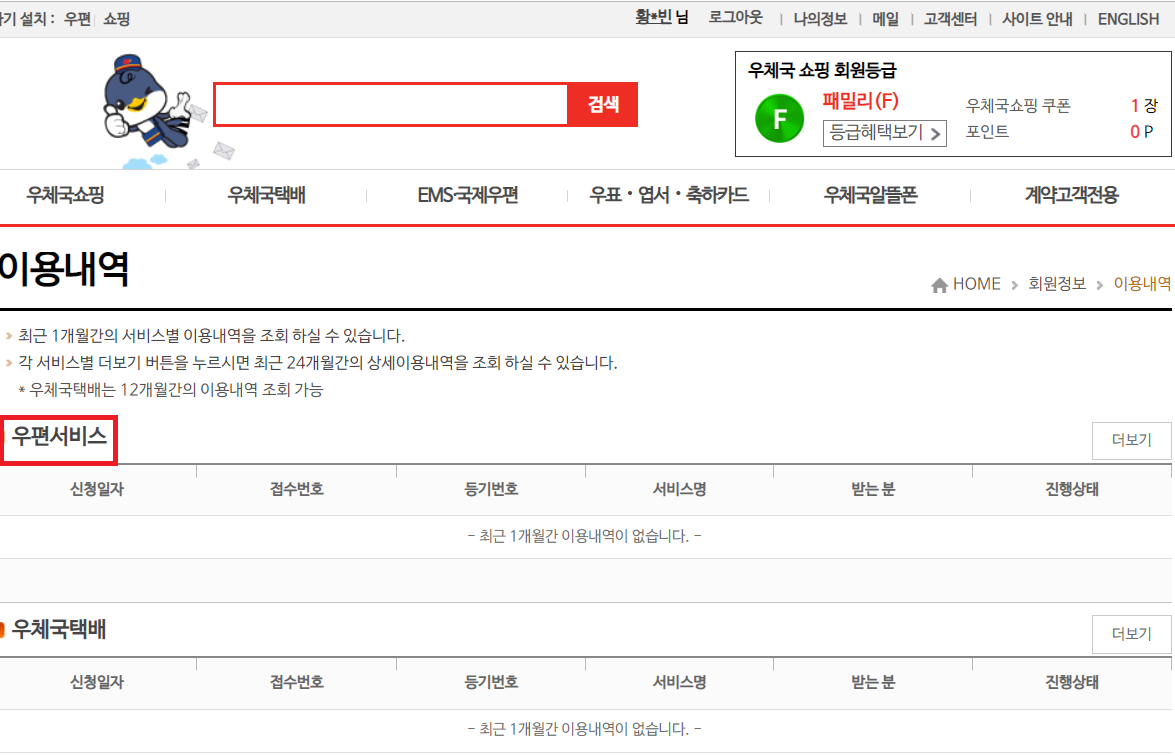
* **ID별 보낸 우편 조회**

질의



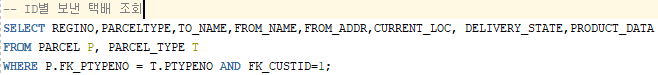
결과



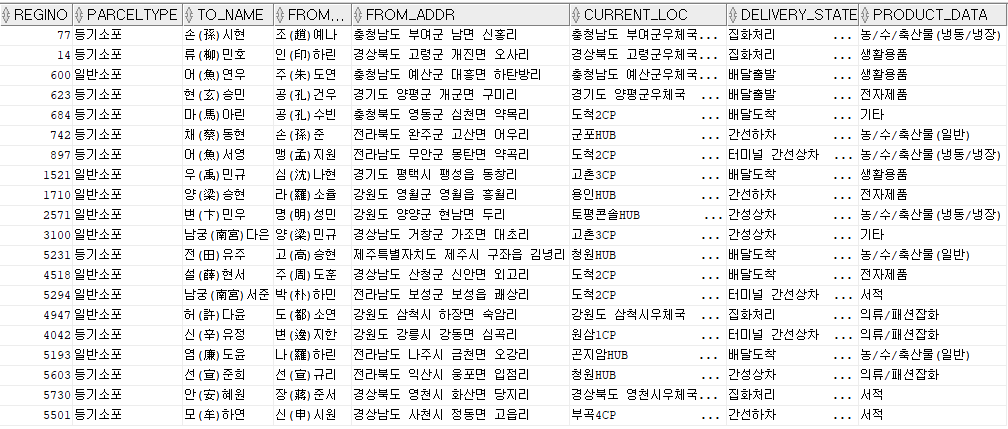


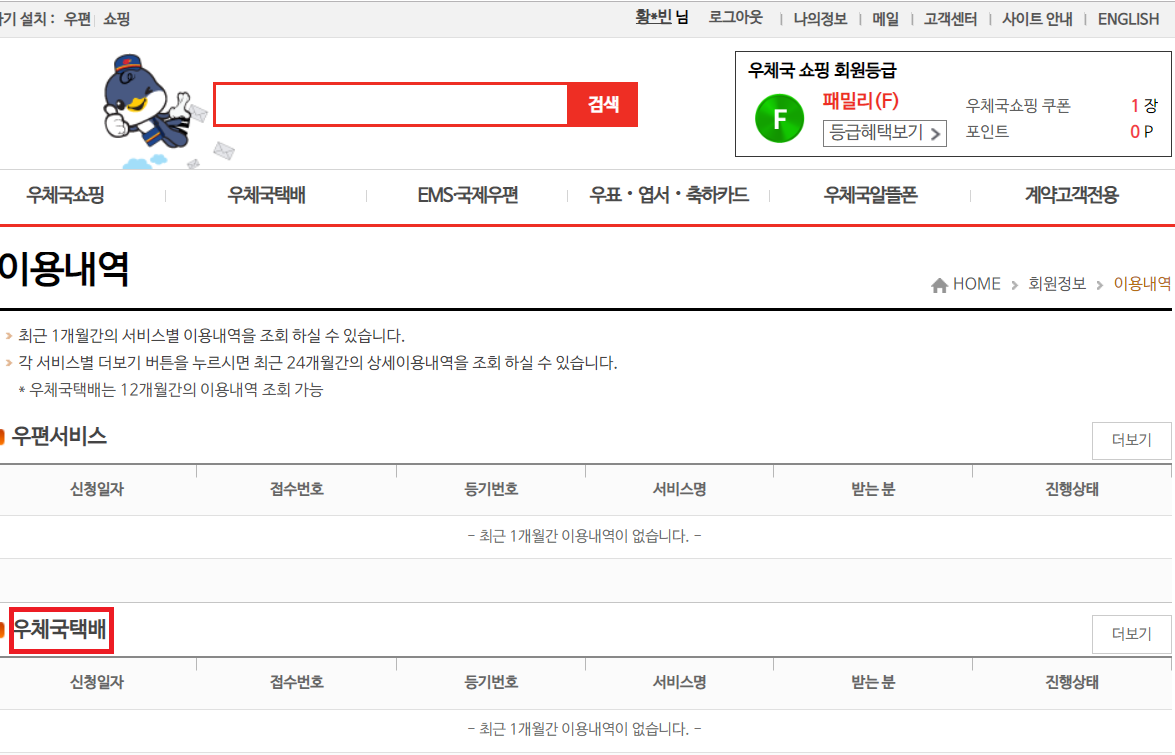
* **ID별 보낸 택배 조회**

질의



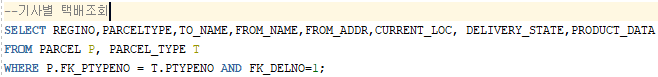
결과



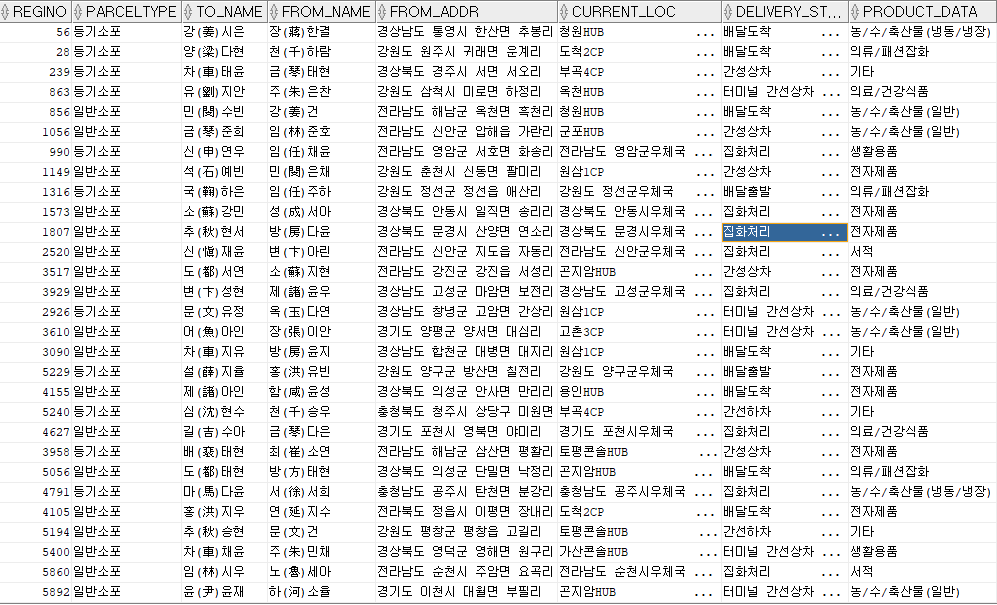


* **기사별 택배조회**

질의

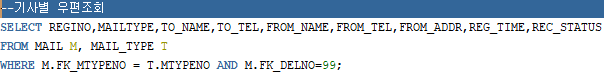


결과

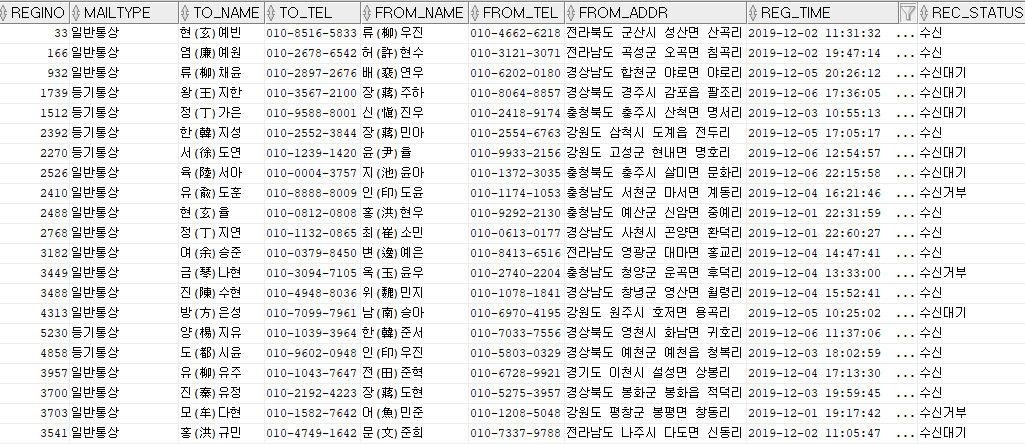


* **기사별 우편조회**

질의

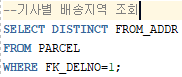


결과

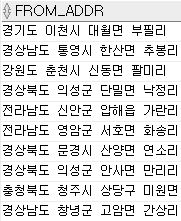
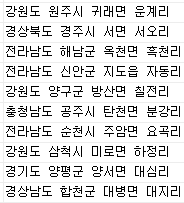
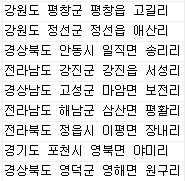
****

* **기사별 배송지역 조회**

질의

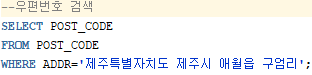


결과

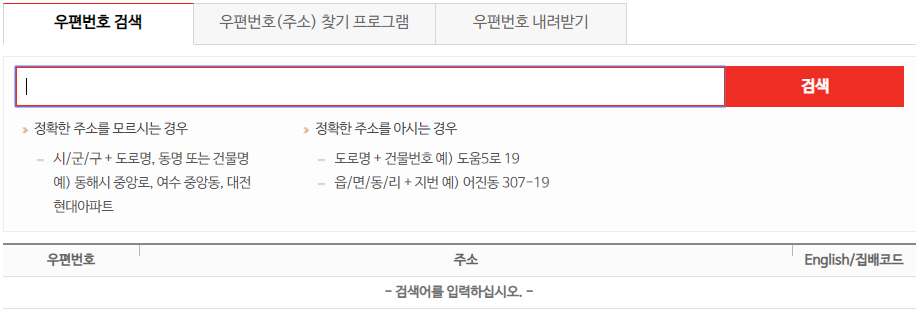
* **우편번호 검색**

질의



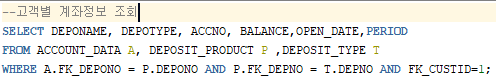
결과

****

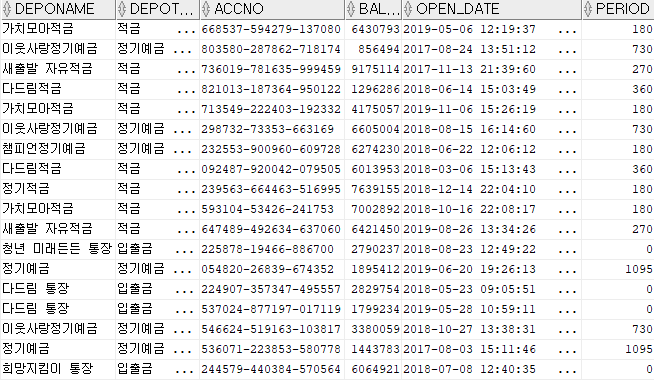
****

* **고객별 계좌정보 조회**

질의

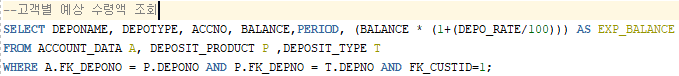


결과

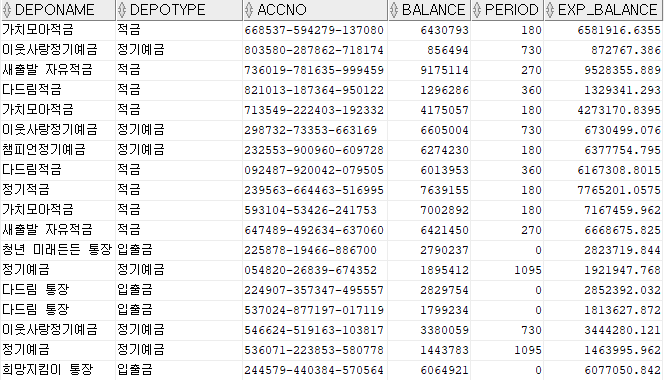


* **고객별 예상 수령액 조회**

질의

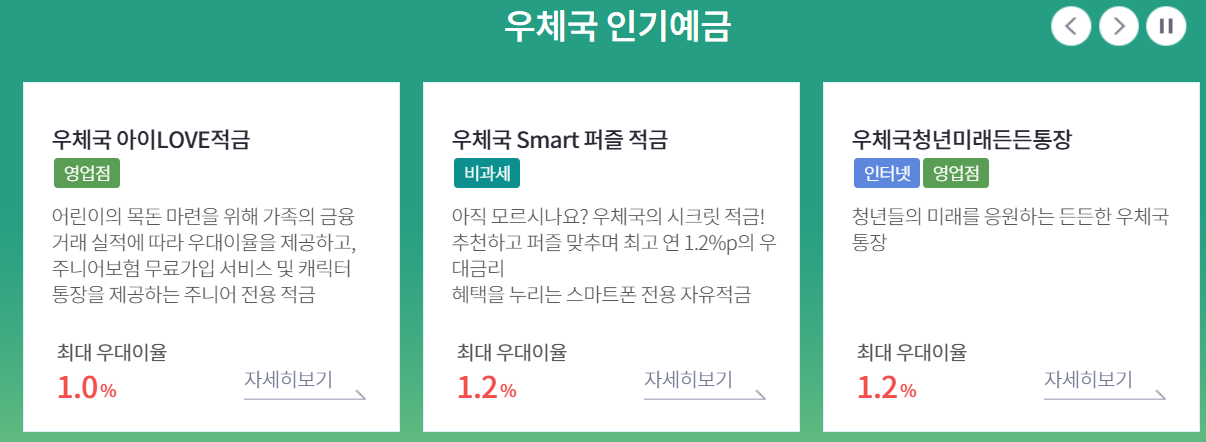
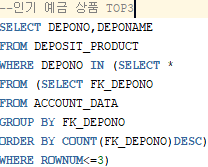


결과



* **인기 예금 상품 TOP3**

질의

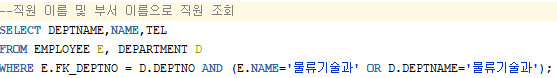
****

결과

****

* **직원 이름 및 부서 이름으로 직원 조회**

질의



결과

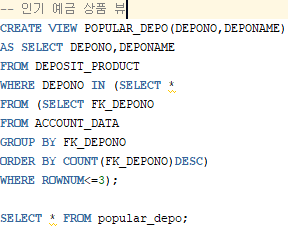




* **뷰 생성**
* **인기 예금 상품의 뷰**

-> 질의문 작성이 어렵기 때문에 뷰를 만들어 사용했습니다.

질의



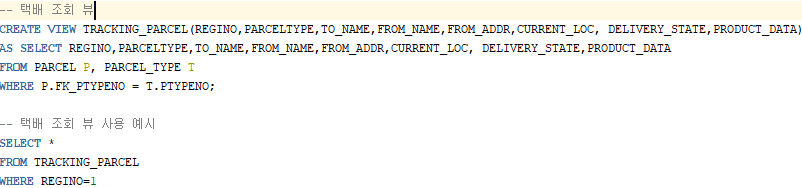
결과



* **택배 조회 뷰**

-> 질의문 작성이 어렵고 자주 사용하기 때문에 뷰를 만들어 사용했습니다.

질의



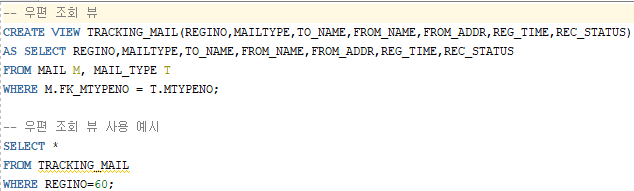
결과



* **우편 조회 뷰**

-> 질의문 작성이 어렵고 자주 사용하기 때문에 뷰를 만들어 사용했습니다.

질의



결과



**6. 기타**

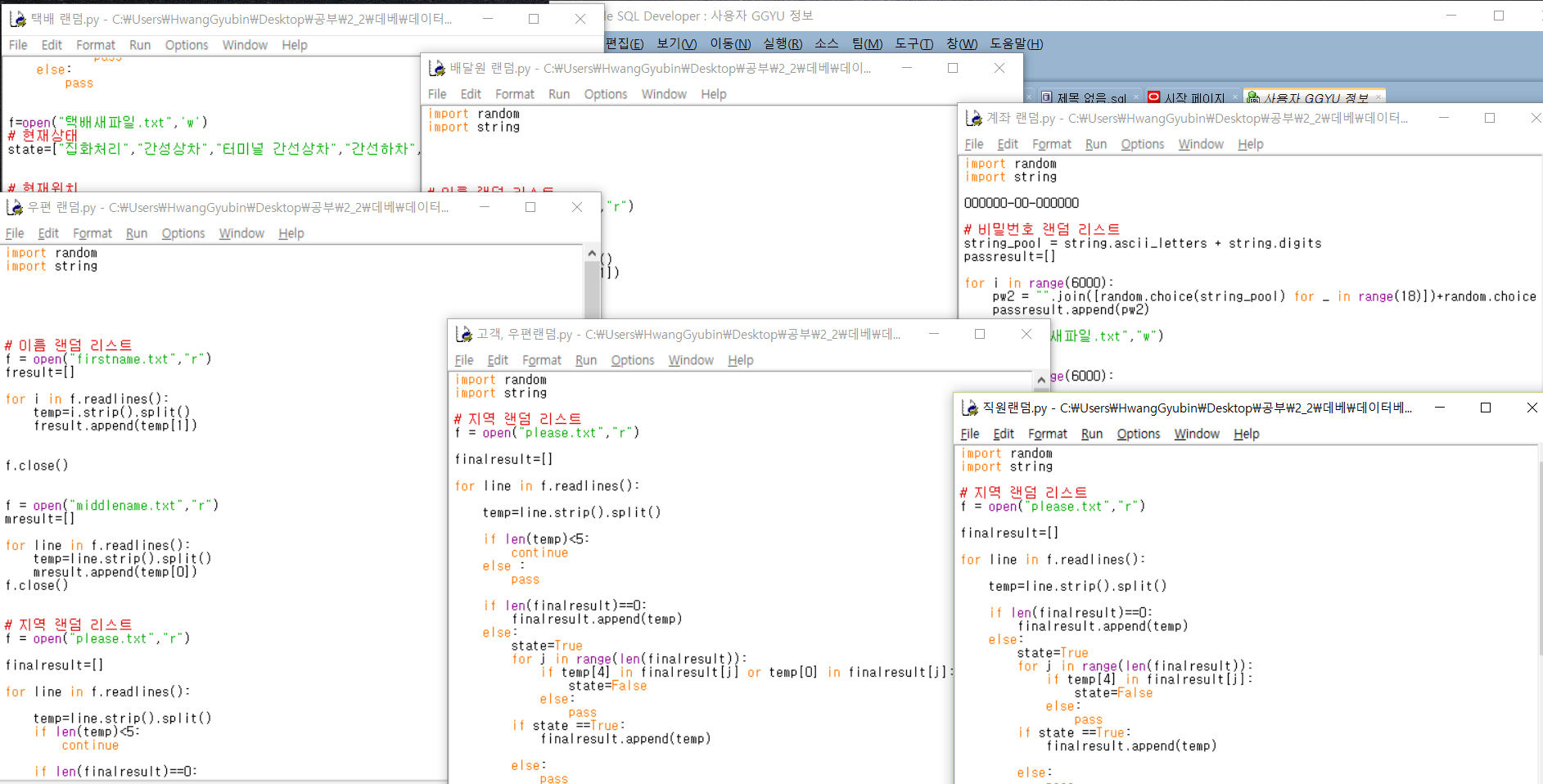
* **데이터를 수집한 방법과 이유**

인덱스를 사용하는 예를 들기 위해 많은 데이터가 필요하다고 생각했고, 랜덤의 데이터를 만들어 인덱스를 부여해 시간을 비교해 봤습니다.

**우편 번호**는 우체국에서 제공하는 파일을 받아, 파이썬을 사용하여 우편번호와 지역명을 선별하여 약 1000개의 데이터를 받았고, 그 데이터를 데이터베이스에 입력하였습니다.



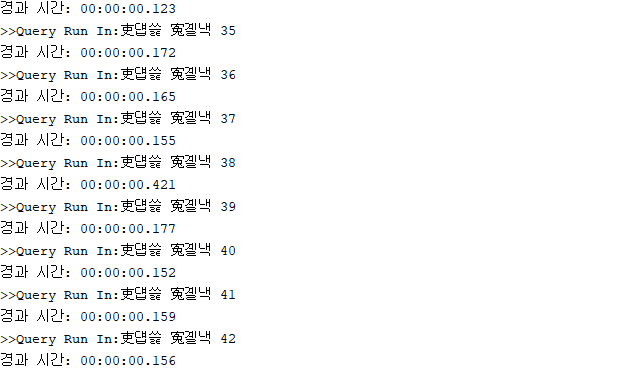
**택배, 우편, 배달원, 고객, 계좌, 직원**의 데이터는 파이썬을 사용하여 랜덤으로 데이터를 생성하였습니다.



처음 PARCEL 릴레이션에6000개의 데이터를 삽입하고 CUSTID 애트리뷰트에 보조 인덱스를 부여했을 때 실행시간의 차이가 거의 차이가 없었고, 이후 7만개를 시도를 했지만 차이가 없었습니다.

그래서 저희가 내린 결론은 현재 상황이 릴레이션의 크기가 작아 인덱스가 도움이 되지 않는 상황이라고 판단하고 인덱스를 사용하지 않기로 했습니다.

**- 인덱스 적용 전**



**- 인덱스 적용 후**



* **시행착오**

**- 첫번째 어려움**

정규화 과정에서 BCNF까지 정규화를 마쳤음에도 불구하고, 갱신이상이 여전히 지속되는 모습이 보였다.

해결 및 느낀 점

이를 인터넷 검색 및 교수님께 도움을 청하여 해결하고자 하였다.

ER 다이어그램에서 개념적 설계부터 잘못되어서 갱신이상이 생기고, 정규화 과정은 모든 갱신이상을 해결하는 것이 아닌, 그 수를 줄여주는 것이라는 점을 알게 되었다

**- 두번째 어려움**

우체국 제공 DB 반영하는 것, 정식 홈페이지에서 제공하는 만큼 그 크기가 무척이나 크고 방대했고,

이를 저희가 원하는 형식으로 사용하고자 바꾸는 과정에서 어려움이 많았다

해결

PYTHON 프로그래밍을 통해 많은 양의 데이터를 보다 편하고, 정확하게 사용할 수 있도록 했다

느낀 점

적으면 적다고 말할 수 있는 양의 데이터를 변형하고 이용하는 과정도 이렇게 많은 어려움을 겪었는데, 실제 현실에서 사용되는 데이터베이스들을 설계하고 운영하기 위해선 정말 많은 문제를 겪고 해결해야 할 것 같다

**- 세번째 어려움**

데이터를 넣을 때, 많은 수의 데이터를 넣는 만큼 많은 제약조건들을 신경 써야 했고, 저희가 고안한 ER스키마에 따른 관계는 없지만, 실질적으로 데이터를 넣는 상황에선 이어져야 하는 경우도 많았고, 반대로 실세계에선 연관이 있는 정보들이, 데이터베이스 질의를 위해선 연결되지 않는 경우가 좋은 경우도 많았다.

해결

처음 ER스키마를 철저하게 고안하고자 했다

느낀 점

더 나은 데이터 베이스를 설계할 때, 생길 수 있는 문제를 ER 단계에서 처음부터 고안하고 대처하기는 힘들고, 후에 데이터 삽입에서 알 수 있을 것 같다

그리고 설계과정을 반복해야, 좋은 데이터 베이스 설계가 될 수 있을 것 같다