데이터 베이스 시스템 01분반

프로젝트

소프트웨어학부

20194653

윤다인

**1. 개발 환경 및 구조**

개발 환경 – 파이썬 3.9, visual studio code 사용

main.py – 여기서 모든 API를 불러와 처리한다.

structure.py – 사용한 CLASS를 정의하였다. VLR, SLP, dataDict

API.py – 기능에 대한 함수를 정의하였다.

구조구현의 예시는 모두 아래의 형태를 띈다.

텍스트이(가) 표시된 사진

자동 생성된 설명

table\_name 은 string으로, col\_names 와 col\_types는 같은 length의 lsit로 이루어져 있다.

자세한 구현 결과값은 4. 결과 에서 확인 가능하다.

**2. structure.py 필요한 클래스를 저장한 파일**

**1) variable length record (VLR)**

구현하고자 한 VLR의 구조는 다음과 같다

|  |
| --- |
| Null 이 없을 경우 |
|  |
| Null 이 있는 경우 - 아예 공간을 차지하지 않도록 하였다. Null bitmap의 값만 바꾸어주었다. |
|  |

|  |  |
| --- | --- |
|  | VLR의 최대 길이는 42 bytes로 정하였다.  VLR class에 들어있는 정보는 bytearray 로 구현된 vlr 이 vlr 이 포함된 table명, null bitmap, 해당 column의 값이 null인지 아닌지 판단하는 isNotNull이 있다.  colType, colName, value 의 값은 초기화 할 때는 쓰이지 않고 record 검색할 때 쓰인다. |
|  | makeVLR은 record 의 값(string list)을 bytearray로 바꾸는 메서드이다.  Null bitmap을 만드는 함수는 아래의 checkNull을 사용하였다.  여기서 반환된 정수값을 1byte로 바꿔 [0:1]위치에 1byte로 저장한다.  numNeed는 가변길이 정보를 저장하는 시작 위치, 저장할 각 value의 길이를 저장한다. |
|  | Null 값을 check 하여 null 인 경우는 공간을 차지하지 않게 하였다.  Type이 c인 값은 그대로 그대로 저장하고 type이 v 인 값은 offset에 저장할 위치, 길이 를 저장한 후 뒤에 길이만큼 따로 저장된다 |

|  |  |
| --- | --- |
|  | makeVLR\_bytearray는 SLP에서 추출한 bytearray의 정보를 기반으로 그 record의 값(string list) 로 바꾸는 makeVLR과 정반대의 기능을 하는 메소드이다. |
|  | checkNull은 makeVLR 메서드에서 쓰이는 메서드이다. 1 바이트(0을 8개)로 초기화 한 후 null 값이 있으면 1로 바꾼다. 그리고 이를 다시 int 로 바꾼 값과, 해당 column의 값이 null 인지 아닌지 T/F 로 Boolean list를 return 한다. |
|  | 해당 VLR의 정보를 출력하는 함수이다 |

**2) slotted page (SLP)**

구현하고자 하는 SLP의 구조는 다음과 같다

|  |
| --- |
| 초기화 |
|  |
| Record가 삽입된 경우 |
|  |

|  |
| --- |
|  |
| 최대 길이는 100bytes 로 정하였다.  각 헤더의 크기는 3으로 정하였다. (위 구조 사진에서 확인 가능) |
|  |
| SLP의 정보를 가져오는 메소드이다. Table\_name 과 slotNum 는 테이블의 몇번째 slot 의 정보를 불러올지에 대한 인자이다 |
|  |
| SLP의 정보를 출력하는 메소드이다 |

3) Data Dictionary

|  |  |
| --- | --- |
|  | 초기화 부분이다.  저장될 table 이름, 컬럼명, 각 컬럼별 타입(varchar인지 char인지) 테이블의 slot 개수, 테이블의 record개수가 저장된다. |
|  | Table name, col name, col type을 입력값으로 받아 write 한다.  이 메서드는 createTable에서 실행된다. 하나의 테이블은 하나의 폴더를 생성하고 하나의 폴더 안에 하나의 메타데이터가 하나의 txt로 저장된다 |
|  | 메타데이터의 정보를 출력하는 함수이다. |
|  | dataDict()의 값을 불러오는 메서드이다.  해당 table의 metadata의 값을 가져온다. |
|  | SLP에 변화가 생겼을 때 실행되는 메서드이다.  updateSlot의 값이 True면 slotNum을 +1 하여 write 한다.  updateRecord가 True면 recordNum을 +1 하여 write한다 |

**3. API.py 기능 파일**

**1) createTable – 테이블을 생성**

텍스트, 장치, 오렌지, 측정기이(가) 표시된 사진

자동 생성된 설명

Tabe name, col\_name, col type을 인자로 받아서 table을 생성한다. 인자의 정보들을 받아 data dictionary를 생성한다. 테이블을 처음 생성할 때는 SLP를 만들지 않게 했다.

**2) insertColumn – record 삽입**

|  |
| --- |
|  |
| Table의 metadata를 가져와 해당 폴더에 SLP의 개수를 가져온다. 그리고 record를 vlr 형태로 만든다.  SLP의 개수가 0이면 새로 SLP를 만든 후 값을 넣는다. 새로 만드니 record개수는 1개가 되고 위의 class 의 SLP 의 구조대로 차례차례 정보를 삽입한다.  이 경우 SLP의 개수도 1개가 되고 record의 개수도 1개 늘었으므로 updateSlot과 updateRecord가 모두 True 값이 된다. |
|  |
| 그 테이블(폴더)에 SLP가 하나라도 있는 경우이다. Slotnum을 받아와 가장 마지막 slot을 연다.그 slot에 새로운 record를 넣을 공간이 있는지 확인한다.  만약 모자라면 새로운 SLP를 하나 생성하여 위와 같은 경우(SLP가 0인 경우)를 반복한다. 이 경우 또한 updateSlot과 updateRecord 값이 모두 True이다.  Record가 들어갈 길이가 된다면, 마지막 slot에 삽입한다. 이 경우 slot의 개수는 늘어나지 않으므로 updateSlot은 False, updateRecord의 값만 True가 된다. |
|  |
| 새로운 SLP를 테이블(폴더)에 write 한다.  updateSlot과 updateRecord 값을 가지고 dataDict 메서드의 updateDict을 실행시켜 dataDict 또한 update 된다. |

**3) findRecord – 조건에 맞는 record 검색**

|  |  |
| --- | --- |
|  | Select 은 보고자 하는 column 값, table\_name은 보고자 하는 table, where =target일 경우를 뜻하는 것이다.  최종 결과값으로 result\_vlr 의 array를 return 한다 |
|  | (outer for loop) 해당 table의 slp를 모두 확인한다.  (inner for loop)각 slp에서 record의 정보를 가져온다.  Where=target 조건을 check 한다  만약 select 의 값이 \* 면 record 전체를 append 하고 아니면 select 에 맞는 값을 append 한다 |

**4) findColumn – 해당 table의 column들 검색**

텍스트이(가) 표시된 사진

자동 생성된 설명

Tablename을 인자로 받으면 그 table에 있는 컬럼명을 다 반환한다

**4. 결과**

텍스트이(가) 표시된 사진

자동 생성된 설명

table\_name: table 이름

col\_names: table의 column명

col\_types: table의 column들의 각 type

(c는 고정길이, v는 가변길이이다. c8인 경우, 길이가 8인 고정길이의 값을 가진다. v16은 최대 16길이를 가진다)

**실행 방법**

main.py 파일에서 table\_names, col\_names, col\_types 를 설정한다.

그리고 아래의 코드를 실행한다. (inserRecord 는 insert\_record의 list 필요)

createTable(table\_name,col\_names,col\_types)

insert\_record = ['20194653', 'dain', '1', 'sci']

insertRecord(table\_name, insert\_record)

res = findRecord(select = '\*',

                    table\_name = table\_name,

                    where = 'dept',

                    target = 'sci')

    if (len(res) == 0) : print('query not found')

print(findColumn(table\_name))

**1) table creation**

하나의 테이블 당 하나의 폴더와 meta\_data.txt 생성

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 전자기기, 디스플레이이(가) 표시된 사진

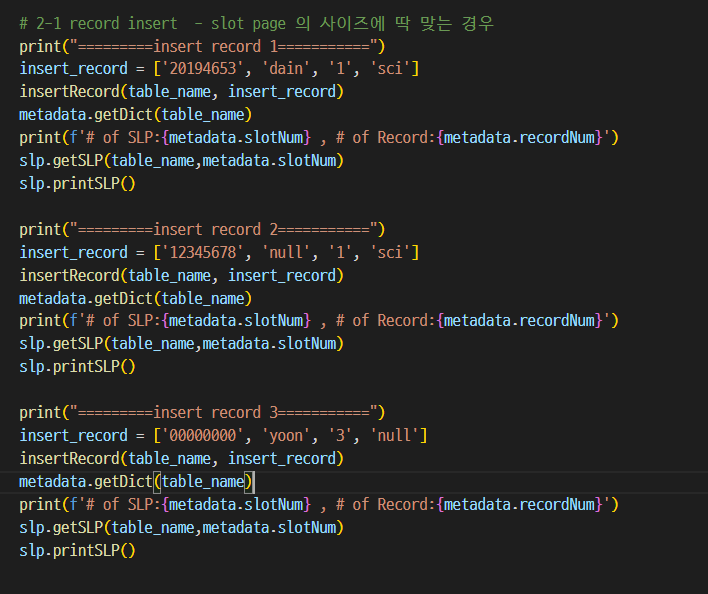
자동 생성된 설명

텍스트, 스크린샷, 모니터, 검은색이(가) 표시된 사진

자동 생성된 설명

폴더와 meta\_data.txt 파일이 생성되었다

**2) record insertion**



텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

Slot 값이 늘어난 것을 확인할 수 있다

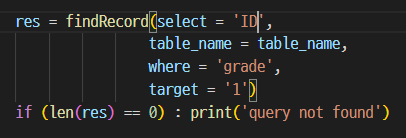
텍스트, 스크린샷, 모니터, 화면이(가) 표시된 사진

자동 생성된 설명

Bin 파일(slot)이 늘어난 것과 meta\_data.txt 의 값이 변경되었다.

**3) search record**

특정 col을 출력



텍스트이(가) 표시된 사진

자동 생성된 설명

전체를 출력

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트, 실내, 스크린샷, 닫기이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

**4) table columns**





**5) 한 번에 실행**

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

**6) 다른 예시 : people 테이블**

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명people 생성

텍스트이(가) 표시된 사진

자동 생성된 설명

meta\_data.txt 생성

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

Meta\_data.txt 값이 업데이트. Slot 수와 record 수 update.

