Simple Storage Manager Implementation

CMPE 321 - Project 2

August 31, 2020

Ramiz Dündar
2016400012

1 Introduction

A storage manager is part of DBMS(Database Management System). It's duty is performing necessary tasks specified in DDL(Data Definition Language) and DML(Data Manipulation Language) on lower levels of a computer system. In this project we are expected to implement a storage manager with simple hierarchy and ability to perform specified DDL and DML tasks. Please also note that this implementation does not include error checking system.

2 Assumptions

- Page size is 1K(112B for testing puposes)
- Page header include: pageID, nRecs, nPage
- Record header include: recID, isEmpty
- Max length of type name is 9 byte
- Max length of field name and field is 10 byte
- A record has maximum 9 fields
- Name of system catalogue is syscat
- Data files named as "typeName" where typeName is name of the record type

3 Storage Structures

3.1 System Catalogue

System catalogue is the unique file holds information about structure about files in the DBMS. Each page can hold 10 records (mrecx100 + 12 < psize)

- Page Header (12 byte)
  - pageID (4 byte)
  - nRecs (4 byte)
  - nPage (4 byte)
- Records (100 byte)
  - typeName (14 byte)
  - isEmpty (1 byte)
  - nFields (4 byte)
  - fieldName (9 byte x 9)

PageID		nRecs		nPage		
typeName 1	isEmpty	nFields	fieldName 1	fieldName 2	...	fieldName 9
typeName 2	isEmpty	nFields	fieldName 1	fieldName 2	...	fieldName 9
...	...	...	...	...	...	...
typeName mrec	isEmpty	nFields	fieldName 1	fieldName 2	...	fieldName 9

3.2 Data Files

A data file is group of same type records. Note that each field is integer. Each page can hold 24 records (24xmrec + 12 < psize)

- Records (m byte)
  - Record Header (5 byte)
    - recID (4 byte)
    - isEmpty (1 byte)
  - field (4 byte x fnumber)
- Page Header (12 byte)
  - pageID (4 byte)
  - nRecs (4 byte)
  - nPage (4 byte)

PageID		nRecs		nPage	
recID 1	isEmpty	field 1	field 2	...	field 9
recID 2	isEmpty	field 1	field 2	...	field 9
...	...	...	...	...	...
recID mrec	isEmpty	field 1	field 2	...	field 9

4 Operations

4.1 DDL

4.1.1 Create a type

```
def create_type():
    rectype = b''

    isempty = 0
    print('type name: ')
    tname = input()

    rectype += encode(tname, 14)
    rectype += encode(isempty, 1)

    print('please enter field names')
    print('maximum 9 fields')
    print('to finish give empty string')
    print('empty string will not create a field')

    nfields = 0
    fields = []
    for i in range(1, 10):
        print(f'please enter field name {i}: ')
        field = input()
        if field == '': break
        fields.append(field)
        nfields += 1

    rectype += encode(nfields)
    for field in fields:
        rectype += encode(field, 9)

    rectype += b'\x00'*((9-len(fields))*9)

    insert_rec('syscat', rectype)

    create_file(tname)
    flush_page(tname, 0)
```

4.1.2 Delete a type

```
def delete_type():
    print('please enter name of the type you want to delete: ')
    rectype = input()
    address = search_rec(sc, rectype)
    f = bytearray(dmr(sc, address[0]))
    f[address[1] + 14:address[1] + 15] = encode(1, 1)
    f[4:8] = encode(decode(f[4:8], int)-1)
    dmw(sc, address[0], f)
    os.remove(db+'/'+rectype)
```

4.1.3 List all types

```
def list_types():
    mrec = find_mrec(sc)
    pnumber = 0
    f = dmr(sc, pnumber)
    reclen = find_reclen(sc)
    hasnext = True

    types = []
    while hasnext:
        for i in range (mrec):
            if decode(f[26+i*reclen:26+i*reclen+1], int) == 0:
                types.append(decode(f[12+i*reclen:12+i*reclen+14], str))

        if decode(f[8:12], int) == -1:
            hasnext = False
        else:
            pnumber += 1
            f = dmr(sc, pnumber)

    for rectype in types:
        print(rectype)

    return rectype
```

4.2 DML

4.2.1 Create a record

```
def create_record():
    print('please enter the type of record you want to enter: ')
    rectype = input()
    nfields = find_nfields(rectype)

    print('please enter the id of record(must be integer) you want to enter: ')
    rec = encode(int(input()))
    rec += encode(0, 1)

    print(f'please enter {nfields} fields(must be integers): ')

    address_sc = search_rec(sc, rectype)
    scf = dmr(sc, address_sc[0])
    offset = address_sc[1] + 19
    for i in range(1, nfields+1):
        fname = decode(scf[offset+9*(i-1):offset+9*i], str)
        print(f'please enter {fname}: ')
        rec += encode(int(input()))

    insert_rec(rectype, rec)
```

4.2.2 Delete a record

```
def delete_record():
    print('please enter name of the type you want to delete: ')
    rectype = input()
    print('please enter id of the type you want to delete: ')
    recid = int(input())
    address = search_rec(rectype, recid)
    f = bytearray(dmr(rectype, address[0]))
    f[address[1] + 4:address[1] + 5] = encode(1, 1)
    f[4:8] = encode(decode(f[4:8], int)-1)
    dmw(rectype, address[0], f)
```

4.2.3 Search for a record

```
def search_record():
    print('please enter name of the type you want to search: ')
    rectype = input()
    print('please enter id of the type you want to search: ')
    recid = int(input())

    address = search_rec(rectype, recid)
    if address is None:
        print('not found')
        return

    f = dmr(rectype, address[0])
    rec = f[address[1]:address[1] + find_reclen(rectype)]

    rid = decode(rec[0:4], int)
    isempty = decode(rec[4:5], int)
    if isempty == 1: return
    print(f'id: {rid}', end=' || ')

    address_sc = search_rec(sc, rectype)
    scf = dmr(sc, address_sc[0])
    offset = address_sc[1] + 19
    for i in range(find_nfields(rectype)):
        field = decode(rec[5+4*i:5+4*(i+1)], int)
        fname = decode(scf[offset+9*i:offset+9*(i+1)], str)
        print(f'{fname}: {field}', end=' || ')

    print()
```

4.2.4 List all records of a type

```
def list_records():
    print('please enter name of the type you want to list: ')
    rectype = input()
    mrec = find_mrec(rectype)
    pnumber = 0
    f = dmr(rectype, pnumber)
    reclen = find_reclen(rectype)
    hasnext = True

    recs = []
    while hasnext:
        for i in range (mrec):
            if decode(f[16+i*reclen:16+i*reclen+1], int) == 0:
                recs.append(decode(f[12+i*reclen:12+i*reclen+4], int))

        if decode(f[8:12], int) == -1:
            hasnext = False
        else:
            pnumber += 1
            f = dmr(rectype, pnumber)

    print(f'total {len(recs)} records')

    for rec in recs:
        print(rec)

    return rectype
```

5 README

Only requirement is python 3. After downloading package run **python main.py**
If you wish to use my test cases extract the zip file and name folder as the global **db** variable
After running the python code you can interact from CLI. Instructions will be written on the CLI.

6 Conclusions

This is a simplistic storage manager implementation where each record has fixed memory. This causes inefficiency if all fields are not used. Also DDL and DML operations are too few. While it is good for understanding database systems it requires more development in order to function practically.