**Homework 4**

파일처리

2012018109 권현우

과제 목표:

RM Layer API를 이용하여 작성한 연락처 프로그램에서 아래 항목을 분석

1. Insert 기능과 Delete 기능이 이루어지는 동작순서
2. RM Layer API function 설명
3. Insert, Delete function 알고리즘 분석
4. **Insert 기능과 Delete 기능이 이루어지는 동작**
   1. *Insert 기능 동작 순서*
5. int main()에서 rmm.CreateFile(“Contactlist.dat”, recordSize) 호출
6. int main()에서 rmm.OpenFile(“Contactlist.dat”, recordSize) 호출
7. int main()에서 menuDisplay() 호출
8. menuDisplay()의 반환값을 select에 assign
9. switch(select)에 따라 select가 1일 경우 case1: 명령 수행
10. int insertContact(RM\_FileHandle &fh) 호출
11. int insertContact(RM\_FileHandle &fh)

7-1. 구조체 person 초기화

7-2. name, addr, telnum, email을 입력받아 person의 field에 assign

7-3. 구조체 person을 char\*로 type-casting하여 record에 assign

7-4. RM\_FileHandle &fh의 InsertRec(record, rid) 호출

* 1. *Delete 기능 동작 순서*

1. Int main()에서 rmm.CreateFile(“Contactlist.dat”, recordSize) 호출
2. Int main()에서 rmm.OpenFile(“Contactlist.dat”, recordSize) 호출
3. Int main()에서 menuDisplay() 호출
4. menuDisplay()의 반환값 select에 assign
5. switch(select)에 따라 select가 2일 떄 case2 명령 수행
6. int othersContact(RM\_FileHandle &fh, int val) 호출
7. int othersContact(RM\_FileHandle &fh, int val)

7-1. val 값이 2일 때 delete 수행.

7-2. 검색할 record의 name을 입력 받아 person\_name에 assign

7-3. RM\_FileScan fs의 OpenScan(…) 함수를 호출하여 file scan initialization.

7-4. RM\_FileScan fs의 GetNextRec(RM\_Record rec)을 호출하여 조건에 맞는 record를 검색

7-5. RM\_Record rec의 GetRid(RID rid) 호출

7-6. RM\_FileHandle &fh의 DeleteRec(rid) 호출

1. **RM Layer API function 설명**

**# RM\_Manager ::** class내에 variable로 PF\_Manager \*pPfm을 가지고 있다.

1. CreateFile(FILE, recordSize)

: 이름이 FILE인 새로운 RM file을 생성한다.

pPfm->CreateFile(FILE), pPfm->OpenFile(FILE, PF\_FileHandle pfFileHandle)을 호출하여 file을 생성한 뒤 열어준다. pfFileHandle의 AllocatePage(PF\_PageHandle pageHandle)을 호출하여 header page를 할당한 뒤, pageHandle.GetData(char \*pData)를 통해 header 정보를 write할 pointer를 pData에 assign한다. 이후 pData를 RM\_FileHdr\* 로 type-casting하여 RM\_FileHdr 구조체의 field인 firstFree, recordSize, numRecordsPerPage, pageHeaderSize, numRecords를 초기화한다.

1. OpenFile(FILE, fh)

: pPfm->OpenFIle(FILE, fh.pfFileHandle)을 통해서 이름이 FILE인 paged file을 open하고 fh.pfFileHandle에 open된 file을 참조값을 할당한다. 이후 GetFirstPage(PF\_PageHandle)을 통해 header page의 참조값을 pageHandle에 할당한 뒤, pageHandle.GetData(pData)를 통해 header information의 pointer를 pData에 할당한다. 이후 memcpy를 통해 pData의 정보를 fileHandle.filHdr 에 read한 뒤 header page는 unpin한다.

1. CloseFile(fh)

: fh에 연관된 file을 close한다. 만약, fh.bHdrChaged가 true라면 memcpy를 통해 file header를 update한다. 이후 fh의 fileHdr와 firstFree를 reset한다.

1. DestroyFile(FILE)

: pPfm->DestroyFile(FILE)을 호출하여 이름이 FILE인 RM file을 제거한다. FILE은 open되지 않은 상태로 존재하여야한다.

**# RM\_FileHandle ::** PF\_FileHandle pfFileHandle, RM\_FileHdr fileHdr, int bHdrChaged를 class variable로 가짐.

1. GetRec(const RID &rid, RM\_Record &rec)

: rid를 통해 record를 반환한다. rid.GetPageNum(pageNum), rid.GetSlotNum(slotNum)을 통해 rid의 pageNum과 slotNum을 추출하고 GetThisPage(pageNum, pageHandle)을 통해 rid가 가리키는 page를 pageHandle에 저장한다. 이후 pageHandle.GetData(pData)를 통해 page의 내용을 pData에 assign한다. 이후 memcpy(rec.pData, pData + fileHdr.pageHeaderSize + slotNum\*fileHdr.recordSize, fileHdr.recordSize)를 통해 찾고자 하는 record의 slot을 rec.pData에 copy한다.

1. InsertRec(const char \*pRecordData, RID &rid)

: 만약 free page list가 empty 상태라면 새로운 page를 allocate하고 record를 insert하기 위한 과정을 준비한다. 만약 아닌 경우는 fileHdr의 firstFree에 record를 저장한다. GetThisPage를 통해 새로운 record가 insert될 page를 Pin하고, 비어있는 slot을 찾을 때까지 GetBitmap을 통해 확인한다. 이후 저장하려는 slotNum이 fileHdr의 numRecordsPerPage보다 작은지 확인한 뒤 memcpy를 통해 record를 insert한 뒤 bitmap의 bit를 set한다.

1. DeleteRec(RID &rid)

: record 삭제를 위해 rid로부터 pageNum과 slotNum을 추출하고 pageNum이 가리키는 page를 찾은 뒤 해당 page의 bitmap에서 slotNum에 해당하는 bit가 set되어있는지 확인한다. 만약 set 되어있다면 reset상태로 바꾼 뒤 만약 지운 record가 해당 page에 남아있던 마지막 record였다면 page를 dispose한다.

1. UpdateRec(RM\_Record &rec)

: rec의 rid를 가져와서 pageNum, slotNum을 추출하고 rec의 내용을 pRecordData에 assign한다. 이후 해당하는 page의 slot에 rec의 내용인 pRecordData를 copy한다.

1. ForcePages(PageNum pageNum)

: page를 buffer pool에서 disk로 force하는 작업을 한다. 만약 bHdrChanged가 True라면 update를 하고 PF\_FileHandle의 ForcePages()를 호출하여 page들을 force한다.

1. GetBitmap(char \*map, int idx)

: 주어진 idx에 해당하는 map의 bit를 반환한다.

1. SetBitmap(char \*map, int idx)

: 주어진 idx에 해당하는 map의 bit를 set한다.

1. ClrBitmap(char \*map, int idx)

: 주어진 idx에 해당하는 map의 bit를 clear(reset)한다.

**# RM\_Record ::** pData와 recordSize를 class의 variable로 가진다.

1. GetData(char \*&\_pData)

: record의 data를 \_pData로 반환한다.

1. GetRid(RID &\_rid)

: record의 rid를 \_rid로 반환한다.

**# RM\_FileScan**

1. OpenScan(const RM\_FileHandle &fileHandle, AttrType \_attrType, int \_attrLength, int attrOffset, CompOp \_compOp, void \*\_value, ClientHint \_pinHint)

: fileHandle과 scan condition들을 설정하여 file scan을 open한다. \_compOp는 scan을 어떻게 할 것인지, \_value는 주어진 attribute와 비교할 value를 가리키는 pointer를 의미한다. 주어진 parameter들을 RM\_FileScan의 local variable로 copy하고 local state variable들을 set한다.

1. GetNextRec (RM\_Record &rec)

: scan condition을 모두 만족하는 record를 찾아서 rec에 set한다. FindNextRecInCurPage\*를 통해 scan condition에 맞는 record를 찾고, 만약 page의 모든 slot을 retrieve했는데 존재하지 않으면 다음 page를 가져온다. 존재한다면 해당 record를 copy한다.

1. FindNextRecInCurPage(char \*pData)

: 현재 page의 slot들을 반복하면서 scan condition에 맞는 record를 찾는다.

1. CloseScan()

: FileScan이 열려있다면, local variable들을 reset한다.

1. **Insert, Delete function 알고리즘 분석**
2. Insert

저장할 data들을 person 구조체로 받은 뒤, record에 저장한 뒤,

filehandle을 통해 record단위로 insert한다.

1. Delete

삭제할 record를 이름을 통해서 검색한 뒤, 입력한 이름과 동일한 이름의 record가 있는지 검색한다. 이후, record의 rid를 가져와서 해당 rid에 해당하는 record를 제거한다.

1. **Application 분석**
2. Open시, RM file이 존재하지 않으면, recordSize 크기의 고정된 record를 갖는 RM file을 생성한다.
3. File 내에 record 단위로 data를 추가/제거 한다.
4. Record를 추가할 때, bitmap에서 해당 slot에 record가 존재하는지 확인 후, 존재하지 않는 slot에 대해 삽입한다
5. Record를 제거할 때, rid를 통해 pageNum과 slotNum을 찾아 bitmap에서 해당 slot이 set되어있으면 reset으로 한다.