DataBase Project 3 보고서

20180038 박형규

1.서론

Project3인 EduOM Project에서는 오디세우스/COSMOS OM의 기능들 중 Object를 저장하기 위한 slotted page 관련 구조에 대한 연산 일부를 구현하도록 하겠다. 먼저 연산을 구현하는데 사용되는 구조체에 대한 분석을 하겠다. sm_CatOverlayForData 구조체는 데이터 file에 대한 정보를 저장하기 위한 데이터 구조이다. 데이터 file이란 서로 관련 있는 object들이 저장된 page들의 집합인데 이러한 각 데이터 file마다 별도로 정보를 유지한다. SlottedPage는 Object의 효율적인 저장 및 관리를 위한 page를 의미하는 데이터 구조이다. 이 데이터 구조는 page에 대한 정보를 저장하는 page header과 object를 저장하는 data 영역, 그리고 object 식별에 필요한 정보를 저장하는 slot들의 array인 slot으로 구성된다. 다음으로 page내에 저장되는 object를 나타내기 위한 Object 데이터 구조가 있다. 이는 Object에 대한 정보를 저장하는 header과 데이터를 저장하는 data 영역으로 나누어져 있다. 이번 Project에서는 6개의 API Function과 1개의 Internal Function을 구현하게 된다. 구현할 함수에 대한 설명은 아래 2. 함수분석에서 계속하도록 하겠다.

2.함수분석

-EduOM_CreateObject()

이 함수는 File을 구성하는 page들 중 파라미터로 지정한 object와 같은 (또는 인접한) page에 새로운 object를 삽입하고, 삽입된 object의 ID를 반환하는 함수이다. 먼저 objectHdr의 properties와 length를 0으로 초기화해준다. 그 뒤 objHdr이 NULL이 아닐 경우에는 objectHdr.tag를 objHdr->tag로 할당해주고 NULL이라면 0으로 할당해준다. 그리고 eduom_CreatObject() function을 실행시켜서 page에 object를 삽입하고 삽입된 object의 ID를 반환한다. 이 때 반환한 ID값이 음수라면 에러가 발생한 것이므로 에러 출력을 시켜준다.

-EduOM_DestroyObject()

이 함수는 File을 구성하는 page에서 object를 삭제하는 함수이다. 먼저 MAKE_PAGEID macro를 이용하여 pid를 할당해준다. 그 뒤에 BfM_GetTrain() 함수를 이용하여 pid를 buffer에 fixed해주고 om_RemoveFromAvailSpaceList() 함수를 이용하여 삭제할 object가 저장된 page를 현재 available space list에서 삭제한다. 그 뒤에 offset, obj, alignedLen을 받아와서 삭제할 object에 대응하는 slot을 EMPTYSLOT으로 설정해 주었다. 그 후 page header을 갱신하였는데 먼저 삭제할 object에 대응하는 slot이 slot array의 마지막 slot인 경우, slot array의 크기를 하나 줄여주었다. 또한, 삭제한 만큼의 부분이 free가 시작되는 부분과 일치한다면 free를 update해주었고 이외의 경우에는 unused를 update 해 주었다. 그 뒤에 File의 Physical ID 또한 MAKE_PAGEID를 이용하여 할당하여 object를 삭제할 file에 관한 정보를 받아온다. 그 뒤에는 for문을 돌며 EMPTYSLOT이 아닌 slot이 가장 처음 나오는 i를 구하고 i가 만약 apage->header.nSlots와 같다면 삭제된 object가 page의 유일한

object일 것이므로 last를 TRUE로 설정해 주었다. 삭제된 object가 page의 유일한 object이고, 해당 page가 file의 첫 번째 page가 아닌 경우에는 om_FileMapDeletePage() 함수를 이용하여 Page를 file 구성 page들로 이루어진 list에서 삭제하였고 Util_getElementFromPool() 함수를 이용하여 dlPool에서 새로운 dealloc list element 한개를 할당 받아서 할당 받은 element에 deallocate할 page 정보를 저장하였고 이 element를 dealloc list의 첫 번째 element로 삽입하였다. 삭제된 object가 page의 유일한 object가 아니거나, 해당 page가 file의 첫 번째 page인 경우에는 Page를 알맞은 available space list에 삽입하여주었다. 마지막으로는 pFid와 pid에 대해 BfM_FreeTrain()을 시켜 주었다.

-EduOM_CompactPage()
이 함수는 Page의 데이터 영역의 모든 자유 공간이 연속된 하나의 contiguous free area를 형성하도록 object들의 offset을 조정하는 함수이다. 먼저 slotNo가 NIL이 아닐 경우에는 slotNo에 대응하는 object를 제외한 page의 모든 object들을 데이터 영역의 가장 앞부분부터 연속되게 저장해 주었고 slotNo가 NIL일 경우에는 모든 object들을 데이터 영역의 가장 앞부분부터 연속되게 저장해 주었다. 그 후 마지막으로 Page header을 갱신해 주었다.

-EduOM_ReadObject()
이 함수는 Object의 데이터 전체 또는 일부를 읽고, 읽은 데이터에 대한 포인터를 반환하는 함수이다. 먼저 EduOM_common.h에 #DEFINE되어 있는 MAKE_PAGEID를 이용하여 pid를 할당해 주었다. 그 뒤에 제공되는 함수인 BFM_GetTrain 함수를 이용하여 pid를 buffer에 fix해 주었고 이로부터 얻어온 SlottedPage Pointer인 apage를 이용하여 offset과 obj를 할당해 주었다. 그 뒤에 length가 REMAINDER이라는 것은 남은 부분을 모두 읽으라는 뜻이므로 length를 남은 부분의 길이 전체가 되도록 설정해 주었고 memcpy를 이용하여 시작부분인 apage->data + offset + sizeof(obj->header) + start 부터 length만큼 읽도록 하였고 마지막으로 주어진 함수인 BfM_FreeTrain 함수를 이용하여 pid를 free 해주었다.

-EduOM_NextObject()
이 함수는 현재 object의 다음 object ID를 반환하는 함수이다. 먼저 MAKE_PAGEID macro와 BfM_GetTrain() 함수, 그리고 GET_PTR_TO_CATENTRY_FOR_DATA를 이용하여 File의 Physical ID에 관한 정보를 받아왔다. 먼저 curOID가 NULL인 경우에는 file의 첫 번째 page의 slot array 상에서의 첫 번째 object의 ID를 반환하도록 하였다. 이는 i를 0으로 설정하고 volNo와 pageNo를 각각 catEntry->fid.volNo와 catEntry->firstPage로 설정하고 MAKE_PAGEID를 통해 pid에 정보를 할당하고 BfM_GetTrain()을 이용함으로써 이루어냈다. curOID가 NULL이 아닌 경우에는 i를 curOID->slotNo + 1으로 설정하고 volNo와 pageNo를 curOID에 있는 정보를 할당함으로써 이루어냈다. 다음으로는 반복문을 이용하여 대응되는 object를 찾는 작업을 수행하였다. slot을 탐색하여 EMPTYSLOT이 처음으로 아닐 때를 찾는다. 이 찾아진 slot은 탐색한 object의 다음 object를 나타내는 slot 일 것이다. 그 뒤, i != apage->header.nSlots인 경우, 즉 탐색한 object 뒤에 또 다른 object가 해당 page에 존재할 경우에는 그 object를 반환하기 위해 MAKE_OBJECTID()를 통해 nextOID를 할당해 주었다. 이외의 경우에는 해당 page에 탐색한 object 이후의 object가 존재하지 않는 경우이므로 탐색한 object가 page의 마지막 object인 경우에는 다음 page의 첫 번째 object ID를 반환

하였고, 탐색한 object가 file의 마지막 page의 마지막 object인 경우에는 EOS를 반환하였다. 다음으로 offset과 obj, objHdr을 설정해주고 pid와 pFid를 BfM_FreeTrain()을 이용하여 free해준 뒤 종료한다.

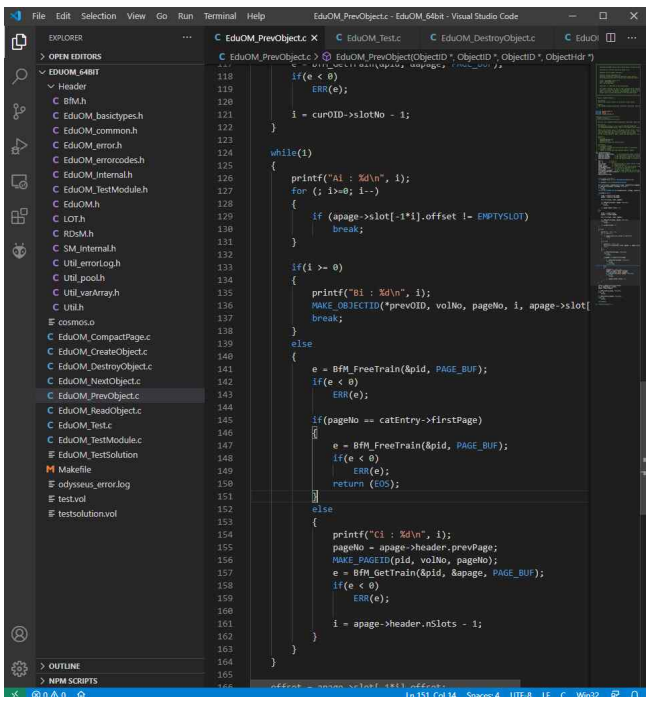
-EduOM_PrevObject()
이 함수는 EduOM_NextObject()와 내용이 거의 비슷하다. 이 함수가 하는 동작은 현재 object의 이전 object의 ID를 반환하는 동작이다. EduOM_NextObject()와 내용 구성도 거의 비슷하므로 달라진 점만 소개하고 넘어가겠다. 먼저 curOID가 NULL인 경우 File의 마지막 page의 slot array 상에서의 마지막 object의 ID를 반환해야 하므로 i가 달라졌고, pageNo 도 catEntry->lastPage로 할당해 주었다. NULL이 아닌 경우에도 i를 curOID->slotNo - 1 로 할당해 주었다. 다음으로 object를 탐색하는 과정에서 EMPTYSLOT이 아닌 slot을 찾을 때 조건에서 i >= 0으로 교체되었고 탐색한 object가 page의 첫 번째 object인 경우와 첫 번째 page의 첫 번째 object인 경우를 고려해야 하므로 catEntry->firstpage를 조건에 삽입하였다.

-eduOM_CreatObject()
이 함수는 internal function으로 file을 구성하는 page들 중 파라미터로 지정한 object와 같은(또는 인접한) page에 새로운 object를 삽입하고, 삽입된 object의 ID를 반환하는 함수이다. 필요한 자유 공간의 크기를 나타내는 변수인 neededSpace에 Object의 삽입을 위해 필요한 자유 공간의 크기를 계산한다. 그 후 MAKE_PAGEID macro와 BfM_GetTrain() 함수, 그리고 GET_PTR_TO_CATENTRY_FOR_DATA를 이용하여 File의 Physical ID에 관한 정보를 받아왔다. 이제 Object를 삽입할 page를 선정해야 하므로 먼저 input 인자로 들어온 nearObj가 NULL이 아닌 경우와 NULL인 경우로 나누었다. 먼저 NULL이 아닌 경우, nearObj가 저장된 page에 여유 공간이 있는지 확인하였다. 여유 공간이 있다면 해당 page를 object를 삽입할 page로 선정하고 선정된 page를 om_RemoveFromAvailSpaceList() 함수를 이용하여 현재 포함되어 있는 available space list에서 삭제하였다. page에 여유 공간은 있지만 compact되어 있지 않은 경우라면 EduOM_CompactPage() 함수를 이용하여 해당 page를 compact 해주었다. 다음으로 nearObj가 저장된 page에 여유 공간이 없는 경우에는 새로운 page를 할당받아서 object를 삽입할 page로 선정하였다. 새로운 page 할당을 위해 RDsM_AllocTrains() 함수를 이용하였고, 그 이후 새롭게 할당한 page의 header을 초기화해 주었다. 그 뒤 om_FileMapAddPage() 함수를 이용하여 선정된 page를 file 구성 page들로 이루어진 list에서 nearObj가 저장된 page의 다음 page로 삽입하였다. 다음으로는 nearObj가 NULL인 경우를 살펴보자. 먼저 object의 삽입을 위해 필요한 free space의 크기에 알맞은 available space list가 존재하는 경우에는 needToAllocPage를 0으로 설정해 주었고, 해당 available space list의 첫 번째 page를 object를 삽입할 page로 선정한다. 그 후 om_RemoveFromAvailableSpaceList() 함수를 이용하여 선정된 page를 현재 available space list에서 삭제하고 여유 공간은 있지만 공간이 compact 되어 있지 않은 경우에는 EduOM_CompactPage() 함수를 이용하여 page를 compact 해 주었다. 다음으로 Object 삽입을 위해 필요한 free space의 크기에 알맞은 available space list가 존재하지 않지만 file의 마지막 page에 여유 공간이 있는 경우가 있다. 이 경우에는 일단 needToAllocPage가 1로 설정되며 file의 마지막 apge를 object를 삽입할 page로 선정하고 여유 공간이 compact

되어 있지 않는 경우에 EduOM_CompactPage() 함수를 이용하여 page를 compact 해 주었다. 마지막 경우로 Object 삽입을 위해 필요한 free space의 크기에 알맞은 available space list가 존재하지 않고 file의 마지막 page에도 여유 공간이 없는 경우가 있을 수 있다. 이 경우에는 새로운 page를 할당 받아서 그 page를 object를 삽입할 page로 선정해야 한다. 따라서 RDsM_AllocTrains() 함수를 통해 page를 할당받고 header을 초기화 하고 om_FileMapAddPage() 함수를 통해 선정된 page를 file의 구성 page들로 이루어진 list에서 마지막 page로 삽입하였다. 이제 page를 선정하였으므로 해당 page에 object를 삽입하면 된다. object의 header에 있는 length 정보를 갱신해 주고, memcpy()를 사용하여 선정한 page의 contiguous free area에 object를 복사하였다. 그리고 slot을 for문을 이용하여 탐색하여 빈 슬롯을 찾은 뒤에 복사한 object의 식별을 위한 정보를 해당 slot에 저장하였고, page의 header에 있는 free를 Object가 새롭게 삽입됨으로써 바뀌기 때문에 갱신해 주었다. 그리고 마지막으로 om_PutInAvailSpaceList() 함수를 통해 Page를 알맞은 available space list에 삽입하고 pFid를 BfM_FreeTrain()을 이용하여 free시켜주며 종료한다.

3.Discussion

이번 과제를 통해 slotted page에서 object를 어떻게 관리하는지 명확히 이해하게 되었다. 특히 코드를 작성하면서 object를 불러올 때 여러 경우를 고려해야 한다는 점을 새롭게 인식하게 되었고 각 경우에 따라 알맞은 연산을 진행해야 한다는 점을 새삼 다시 깨달았다. 또한, 작은 조건이라도 조금만 잘못 생각하면 결과에서 엄청난 차이가 난다는 점도 다시 깨닫게 되었다. 이번 프로젝트를 진행하며 printf를 이용하여 디버깅 하였는데 아래 첨부한 이미지와 같이 i의 값을 알아보기 위하여 printf("Ai : %d\n", i); 등의 printf()문을 활용하여 디버깅을 할 수 있었다.



이렇게 디버깅 한 것과 아래 첨부한 이미지인 ubuntu에서의 결과값을 비교하며 어떤 부분이 잘못된지 확인할 수 있었고 마침내 프로젝트를 성공적으로 마무리 할 수 있었다.

```
Ai : 14
Bi : 14
Ai : 13
Bi : 13
Ai : 12
Bi : 12
Ai : 11
Bi : 11
Ai : 10
Bi : 10
Ai : 9
Bi : 9
Ai : 8
Bi : 8
Ai : 7
Bi : 7
Ai : 6
Bi : 6
Ai : 5
Bi : 5
Ai : 4
Bi : 4
Ai : 3
Bi : 3
Ai : 2
Bi : 2
Ai : 1
Bi : 1
Ai : 0
Bi : 0
Ai : -1
Ci : -1
Ai : 83
Bi : 83
---------------------------------- Result ------------------
+-------------------------------------------------------------+
|                PageID = (1000,  208)                        |
+-------------------------------------------------------------+
+-------------------------------------------------------------+
|  nSlots = 84          free = 3000          unused = 0       |
| FREE = 368            CFREE = 368                           |
+-------------------------------------------------------------+
| fid = (1000,   12)                                          |
| nextPage = 209                prevPage = -1                 |
| spaceListPrev = -1            spaceListNext = -1            |
+-------------------------------------------------------------+
|  0|        29 EduOM_TestModule_OBJECT_NUM_0                 |
|  1|        29 EduOM_TestModule_OBJECT_NUM_1                 |
|  2|        29 EduOM_OBJECT_FOR_COMPACT_TEST                 |
|  3|        29 EduOM_TestModule_OBJECT_NUM_3                 |
|  5|        29 EduOM_TestModule_OBJECT_NUM_5                 |
|  7|        29 EduOM_TestModule_OBJECT_NUM_7                 |
|  9|        29 EduOM_TestModule_OBJECT_NUM_9                 |
| 11|        30 EduOM_TestModule_OBJECT_NUM_11                |
| 13|        30 EduOM_TestModule_OBJECT_NUM_13                |
| 15|        30 EduOM_TestModule_OBJECT_NUM_15                |
| 17|        30 EduOM_TestModule_OBJECT_NUM_17                |
| 19|        30 EduOM_TestModule_OBJECT_NUM_19                |
| 21|        30 EduOM_TestModule_OBJECT_NUM_21                |
| 22|        30 EduOM_TestModule_OBJECT_NUM_22                |
| 23|        30 EduOM_TestModule_OBJECT_NUM_23                |
| 24|        30 EduOM_TestModule_OBJECT_NUM_24                |
| 25|        30 EduOM_TestModule_OBJECT_NUM_25                |
```