

# 오디세우스/EduCOSMOS Project #2: EduOM Project Manual

Version 1.0

Copyright (c) 2013-2015, Kyu-Young Whang, KAIST  
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

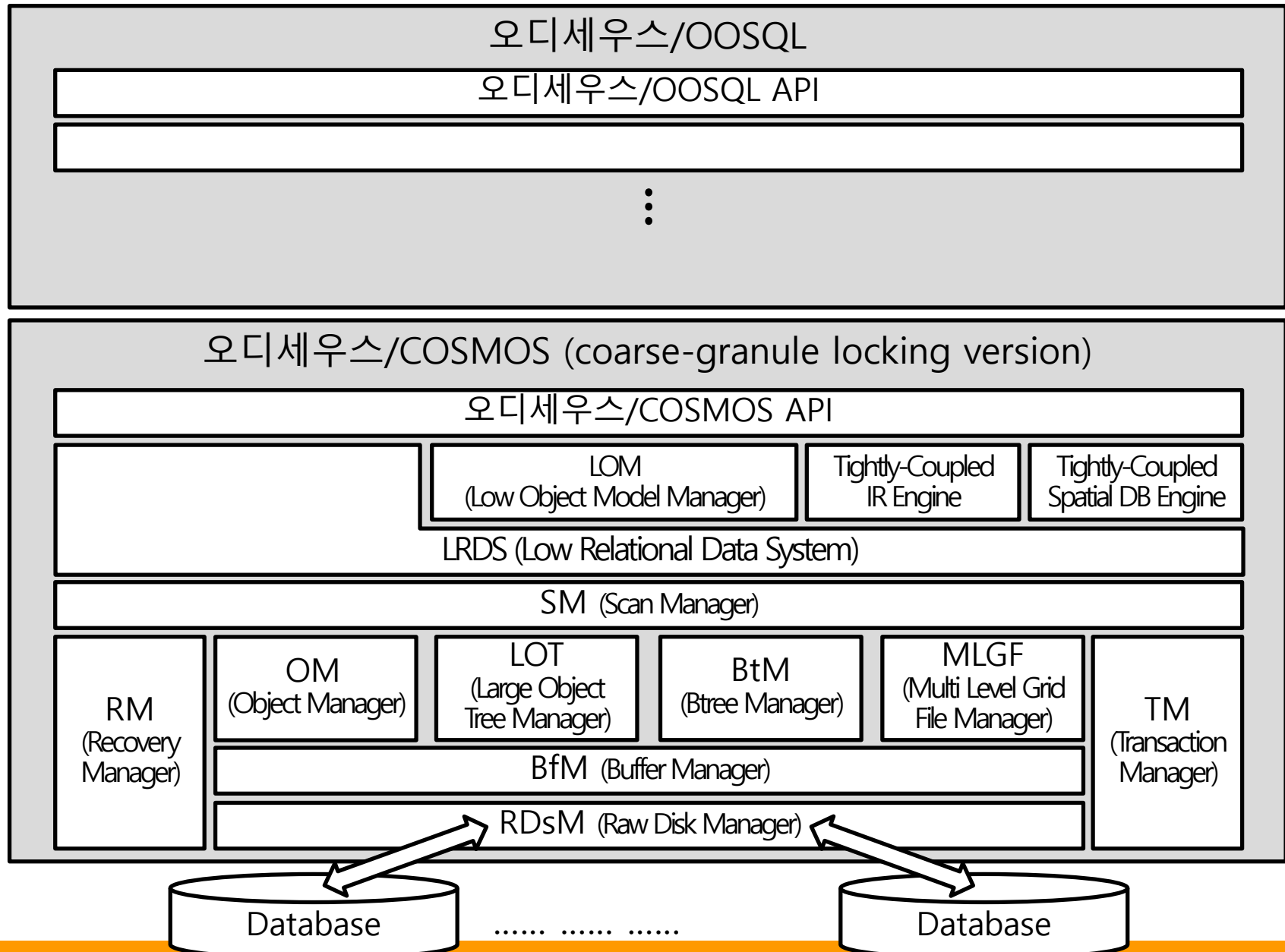
# 목차

- 소개
  - 오디세우스/COSMOS
  - 오디세우스/EduCOSMOS Project
- EduOM Project
  - 자료 구조 및 연산
  - 구현할 Function 들
  - 제공되는 Function 들
  - Error 처리
- Project 수행 방법
- Appendix: Function Call Graph

# 오디세우스/COSMOS

- 오디세우스
  - 1990년부터 황규영 교수 (첨단정보기술 연구센터 (AITrc) / KAIST 전산학과) 등이 개발한 객체 관계형 DBMS
- 오디세우스/COSMOS
  - 오디세우스의 저장 시스템으로서, 각종 데이터베이스 응용 소프트웨어의 하부 구조로 사용되고 있음

# - 오디세우스 구조



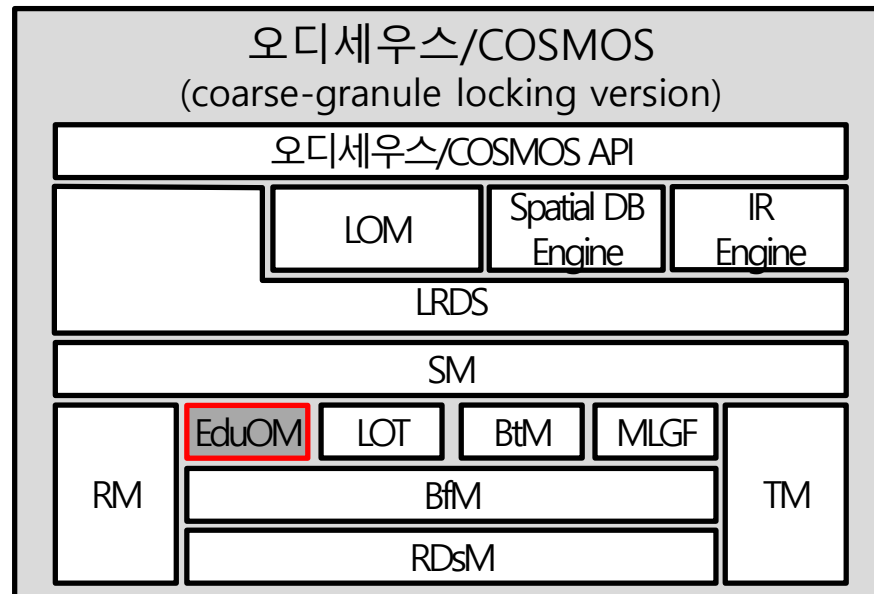
# 오디세우스/EduCOSMOS Project

- 개요
  - Coarse granule locking 버전 오디세우스/COSMOS의 일부분을 구현하는 교육 목적용 project
    - 프로젝트 선행 조건: 기초적인 C 프로그래밍 능력
- 목표
  - 오디세우스/COSMOS의 일부분을 구현함으로써 DBMS 각 모듈별 기능을 학습함
- Project 종류
  - EduBfM
    - Buffer manager에 대한 연산들을 구현함
  - EduOM
    - Object manager와 page 관련 구조에 대한 연산들을 구현함
  - EduBtM
    - B+ tree 색인 manager에 대한 연산들을 구현함

# EduOM Project

- 목표

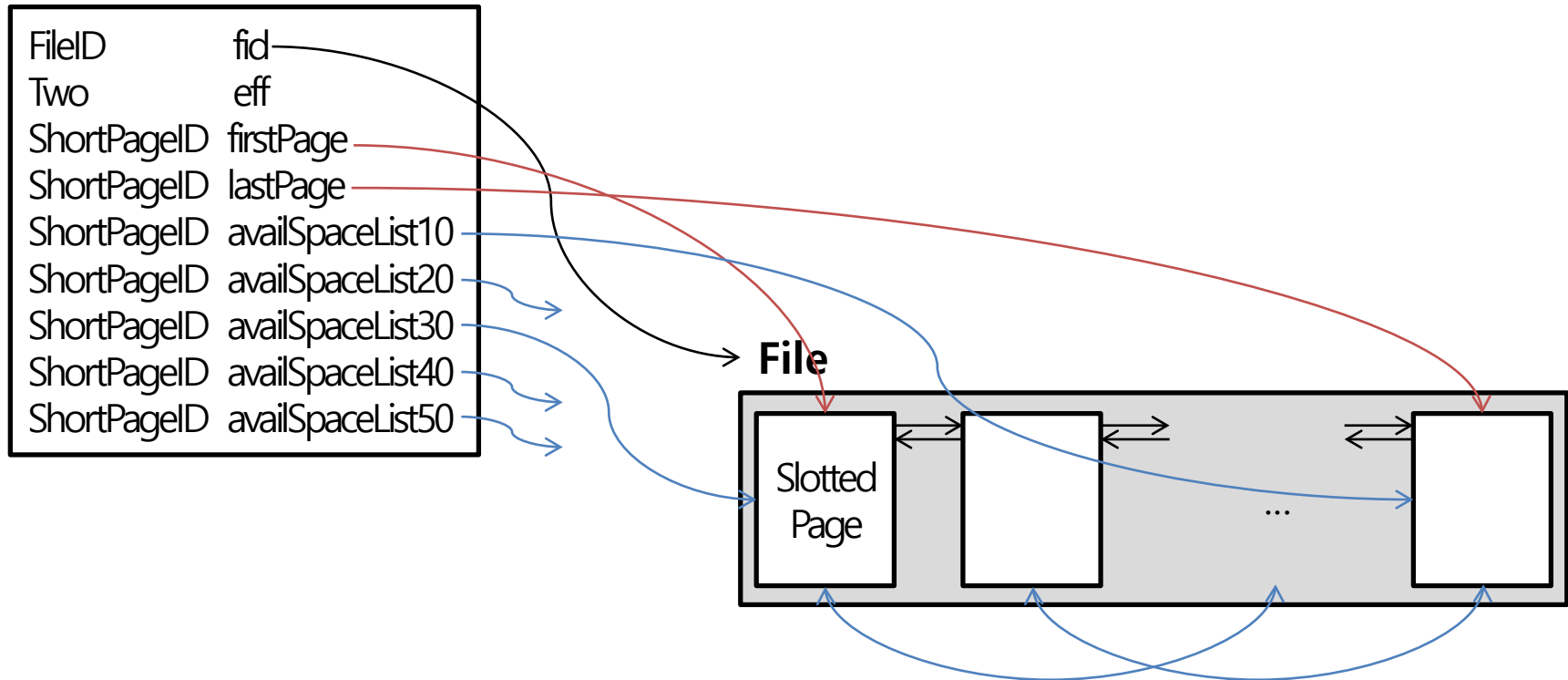
- Object를 저장하기 위한 slotted page 관련 구조에 대한 연산들을 구현함
- EduOM에서는 오디세우스/COSMOS OM의 기능들 중 극히 제한된 일부 기능들만을 구현함



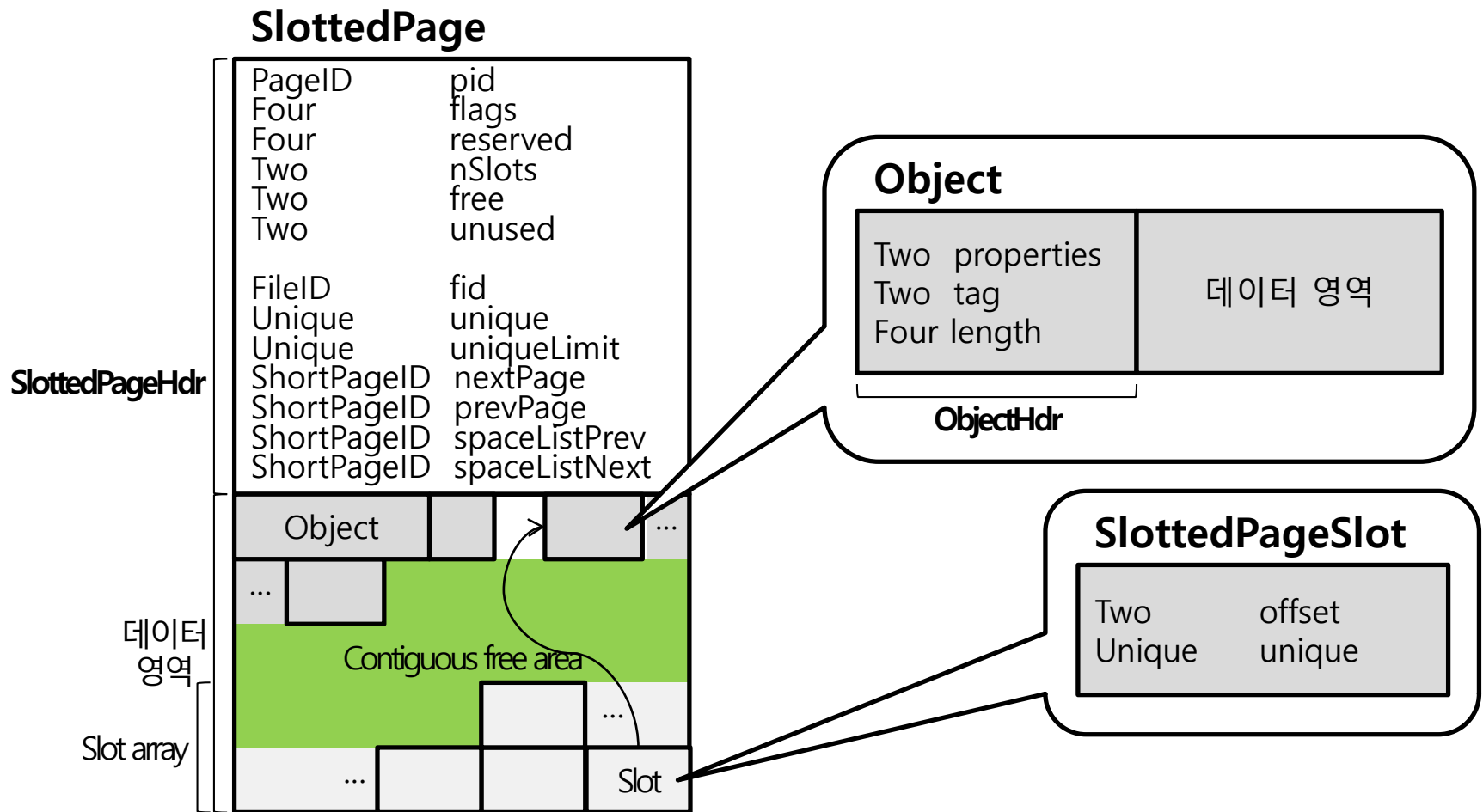
# 데이터 구조



## sm\_CatOverlayForData







# sm\_CatOverlayForData

- 개요
  - 데이터 file에 대한 정보를 저장하기 위한 데이터 구조
    - 데이터 file: 서로 관련 있는 object들이 저장된 page들의 집합
  - 오디세우스/COSMOS의 각 데이터 file 마다 별도로 정보를 유지함
- 구성
  - fid
    - File의 ID
  - eff
    - File의 extent fill factor로서, EduOM에서는 사용하지 않음 (EduOM function 구현 시 이를 무시해도 됨)
  - firstPage
    - File을 구성하는 첫 번째 page의 번호
  - lastPage
    - File을 구성하는 마지막 page의 번호

- availSpaceList10 ~ availSpaceList50
  - 각 available space list를 구성하는 첫 번째 (가장 최근에 삽입된) page의 번호
    - 10% available space list
      - » File을 구성하는 page들 중 자유 공간의 크기가 전체의 10% ~ 20% 인 page들의 doubly linked list
    - 20% available space list
      - » File을 구성하는 page들 중 자유 공간의 크기가 전체의 20% ~ 30% 인 page들의 doubly linked list
    - 30% available space list
      - » File을 구성하는 page들 중 자유 공간의 크기가 전체의 30% ~ 40% 인 page들의 doubly linked list
    - 40% available space list
      - » File을 구성하는 page들 중 자유 공간의 크기가 전체의 40% ~ 50% 인 page들의 doubly linked list
    - 50% available space list
      - » File을 구성하는 page들 중 자유 공간의 크기가 전체의 50% 이상인 page들의 doubly linked list

# SlottedPage

- 개요
  - Object의 효율적인 저장 및 관리를 위한 page 데이터 구조
- 구성
  - header
    - Page에 대한 정보를 저장하는 page header
  - data[]
    - Object들을 저장하는 데이터 영역
  - slot[1]
    - Page에 저장된 object 식별에 필요한 정보를 저장하는 slot들의 array
      - 첫 번째 slot ( $slot[0]$ ) 을 제외한 나머지 slot들 ( $slot[1] \sim slot[n]$ ) 은 데이터 영역과 메모리 공간을 공유함
        - » 첫 번째 slot의 array index = 0
        - » 다음 slot의 array index = 이전 slot의 array index + 1
        - » Slot 번호 = |slot의 array index|

# SlottedPageHdr

- 개요
  - Page에 대한 정보를 저장하기 위한 데이터 구조
- 구성
  - pid
    - Page의 ID
      - Page 번호 및 volume 번호로 구성됨
  - flags
    - Page의 type을 나타내는 bit들의 집합
      - 두 번째 bit가 set 된 경우 (SLOTTED\_PAGE\_TYPE): slotted page임을 나타냄
      - 이외의 bit들은 EduOM에서는 사용하지 않음 (EduOM function 구현시 이를 무시해도 됨)
  - reserved
    - Page에 대한 추가적인 정보를 저장하기 위한 예비 변수

- nSlots
  - Page의 slot array의 크기  
(= 사용중인 slot들 중 마지막 slot의 번호 + 1)
    - Page 내의 효율적인 공간 사용을 위해, object가 삽입/삭제 됨에 따라 slot array의 크기가 동적으로 변화함
- free
  - Page의 데이터 영역 중 contiguous free area의 시작 offset
    - Contiguous free area: 데이터 영역 상에서의 마지막 object 이후의 연속된 자유 공간
- unused
  - Page의 데이터 영역 중 contiguous free area를 제외한 자유 공간들의 크기의 합 (단위: # of bytes)
- fid
  - Page가 속한 file의 ID

- unique
  - Page 내에서 최근 할당된 unique 번호
    - Unique 번호: 각 object마다 할당되는 고유 번호
    - 최초 0으로 초기화하고, om\_GetUnique()를 호출하여 값을 갱신함
- uniqueLimit
  - Page 내에서 현재 할당 가능한 unique 번호의 최대값
    - 최초 0으로 초기화하고, om\_GetUnique()를 호출하여 값을 갱신함
- nextPage / prevPage
  - 같은 file에 속한 다음/이전 page의 번호
    - File 구성 page들간의 double linked list 구조 유지를 위해 사용됨
- spaceListPrev / spaceListNext
  - 같은 available space list에 속한 이전/다음 page의 번호
    - Available space list 구성 page들간의 double linked list 구조 유지를 위해 사용됨

# SlottedPageSlot

- 개요
  - Page에 저장된 object 식별에 필요한 정보를 저장하기 위한 데이터 구조
- 구성
  - offset
    - Page에 저장된 object의 데이터 영역 상에서의 offset
      - 사용하지 않는 slot일 경우, *offset* := EMPTY\_SLOT
  - unique
    - Object의 unique 번호
      - `om_GetUnique()`를 호출하여 unique 번호를 할당 받음

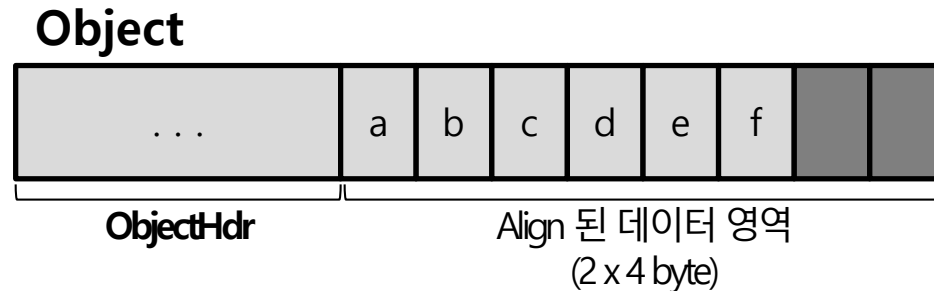


# Object

- 개요
  - Page에 저장되는 object를 나타내는 데이터 구조
  - EduOM에서는 object의 크기 (object header의 크기 + align 된 object 데이터 영역의 크기) 가 page 데이터 영역의 크기보다 작은 small object만을 고려함
- 구성
  - header
    - Object에 대한 정보를 저장하는 object header

– data[]

- Object의 데이터를 저장하는 데이터 영역
- 데이터가 저장된 데이터 영역의 크기는 4의 배수 (32비트 운영체제에서의 메모리 할당 기본 단위) 가 되도록 align 됨
  - 예) 길이가 6인 데이터가 저장된 경우,



# ObjectHdr

- 개요
  - Object에 대한 정보를 저장하기 위한 데이터 구조
- 구성
  - properties
    - Object의 특성을 나타내는 bit들의 집합
      - 모든 bit가 0인 경우, small object임을 나타냄
      - 다른 특성들은 EduOM에서는 사용하지 않음 (EduOM function 구현시 이를 무시해도 됨)
  - tag
    - Object의 tag로서, EduOM에서는 사용하지 않음 (EduOM function 구현시 이를 무시해도 됨)
  - length
    - Object의 데이터의 길이 (단위: # of bytes)
      - Align 된 데이터 영역의 크기가 아닌 데이터 영역에 저장된 데이터의 실제 길이

# 관련 연산

- Page에 새로운 object를 삽입
- Page에서 object를 삭제
- Page의 데이터 영역을 compact
  - 데이터 영역의 모든 자유 공간이 연속된 하나의 contiguous free area를 형성하도록 object들의 offset를 조정함
- Page를 탐색
  - Page에서 원하는 object를 탐색함

# 구현할 API Function 들

- EduOM\_CreateObject()
- EduOM\_DestroyObject()
- EduOM\_CompactPage()
- EduOM\_ReadObject()
- EduOM\_NextObject()
- EduOM\_PrevObject()

(※ API function들은 p.4의 오디세우스/COSMOS API의 일부를 의미함)

(※ API: Application Programming Interface)

# EduOM\_CreateObject()

- 파일: EduOM\_CreateObject.c
- 설명
  - File을 구성하는 page들 중 파라미터로 지정한 object와 같은 (또는 인접한) page에 새로운 object를 삽입하고, 삽입된 object의 ID를 반환함
    - 삽입할 object의 header를 초기화함
      - *properties* := 0x0
      - *length* := 0
      - 파라미터로 주어진 *objHdr*가 NULL이 아닌 경우,
        - » *tag* := *objHdr*에 저장된 tag 값
      - 파라미터로 주어진 *objHdr*가 NULL인 경우,
        - » *tag* := 0
    - `ed uom_CreateObject()`를 호출하여 page에 object를 삽입하고, 삽입된 object의 ID를 반환함

- 파라미터
  - ObjectID \*catObjForFile  
(IN) object를 삽입할 file에 대한 정보 (*sm\_CatOverlayForData*) 가 저장된 object의 ID
  - ObjectID \*nearObj  
(IN) 삽입할 object가 인접 저장 되어야 하는 object의 ID
  - ObjectHdr \*objHdr  
(IN) 삽입할 object의 tag 값이 저장된 object header
  - Four length  
(IN) 삽입할 object의 데이터의 길이 (단위: # of bytes)
  - char \*data  
(IN) 삽입할 object의 데이터
  - ObjectID \*oid  
(OUT) 삽입된 object의 ID
  
- 반환값
  - Four 에러코드
  
- 관련 함수
  - eduum\_CreateObject()

# EduOM\_DestroyObject()

- 파일: EduOM\_DestroyObject.c
- 설명
  - File을 구성하는 page에서 object를 삭제함
    - 삭제할 object가 저장된 page를 현재 available space list에서 삭제함
    - 삭제할 object에 대응하는 slot을 사용하지 않는 빈 slot으로 설정함
      - Slot의 *offset* := EMPTYSLOT
  - Page header를 갱신함
    - 삭제할 object에 대응하는 slot이 slot array의 마지막 slot인 경우, slot array의 크기를 갱신함
      - » *nSlots* := 사용중인 slot들 중 마지막 slot의 번호 + 1
    - 삭제할 object의 데이터 영역 상에서의 offset에 따라 *free* 또는 *unused* 변수를 갱신함



- » Object 삭제로 인해 확보되는 자유 공간의 크기  
 = `sizeof(ObjectHdr)` + align 된 object 데이터 영역의 크기  
 + slot array 크기 갱신에 따라 확보되는 자유 공간의 크기
- 삭제된 object가 page의 유일한 object이고, 해당 page가 file의 첫 번째 page가 아닌 경우,
  - Page를 file 구성 page들로 이루어진 list에서 삭제함
  - 해당 page를 deallocate 함
    - » 파라미터로 주어진 `dIPool`에서 새로운 dealloc list element 한 개를 할당 받음
      - Dealloc list: deallocate 할 page들의 linked list
      - » 할당 받은 element에 deallocate 할 page 정보를 저장함
      - » Deallocate 할 page 정보가 저장된 element를 dealloc list의 첫 번째 element로 삽입함
- 삭제된 object가 page의 유일한 object가 아니거나, 해당 page가 file의 첫 번째 page인 경우,
  - Page를 알맞은 available space list에 삽입함
  - ❖ File의 ID는 그 file의 첫 번째 page의 ID와 동일하며 ID는 불변해야 하므로, 첫 번째 page는 비어있게 되더라도 deallocate하지 않음

- 파라미터

- ObjectID \*catObjForFile  
(IN) object를 삭제할 file에 대한 정보 (*sm\_CatOverlayForData*) 가 저장된 object의 ID
- ObjectID \*oid  
(IN) 삭제할 object의 ID
- Pool \*dlPool  
(INOUT) 새로운 dealloc list element를 할당 받기 위한 pool
- DeallocListElem \*dlHead  
(INOUT) dealloc list의 첫 번째 element를 가리키고 있는 header

- 반환값

- Four 에러코드

- 관련 함수

EduOM\_DestroyObject(), om\_FileMapDeletePage(), om\_PutInAvailSpaceList(),  
om\_RemoveFromAvailSpaceList(), BfM\_GetTrain(), BfM\_FreeTrain(), BfM\_SetDirty(),  
Util\_getElementFromPool()

# EduOM\_CompactPage()

- 파일: EduOM\_CompactPage.c
- 설명
  - Page의 데이터 영역의 모든 자유 공간이 연속된 하나의 contiguous free area를 형성하도록 object들의 offset를 조정함
    - 파라미터로 주어진 *slotNo*가 NIL (-1) 이 아닌 경우,
      - *slotNo*에 대응하는 object를 제외한 page의 모든 object들을 데이터 영역의 가장 앞부분부터 연속되게 저장함
        - » Object 저장 순서: 대응하는 slot 번호 순
      - *slotNo*에 대응하는 object를 데이터 영역 상에서의 마지막 object로 저장함
    - 파라미터로 주어진 *slotNo*가 NIL (-1) 인 경우,
      - Page의 모든 object들을 데이터 영역의 가장 앞부분부터 연속되게 저장함
        - » Object 저장 순서: 대응하는 slot 번호 순
    - Page header를 갱신함

- 파라미터
  - SlottedPage \*apage  
(IN) compact할 page
  - Two slotNo  
(IN) compact할 page의 데이터 영역 상에서 마지막으로  
저장할 object에 대응하는 slot 번호
- 반환값
  - Four 에러코드
- 관련 함수 없음

# EduOM\_ReadObject()

- 파일: EduOM\_ReadObject.c
- 설명
  - Object의 데이터 전체 또는 일부를 읽고, 읽은 데이터에 대한 포인터를 반환함
    - 파라미터로 주어진 *oid*를 이용하여 object에 접근함
    - 파라미터로 주어진 *start* 및 *length*를 고려하여 접근한 object의 데이터를 읽음
      - Object의 데이터 영역 상에서 *start*에 대응하는 offset에서 부터 *length* 만큼의 데이터를 읽음
      - *length*가 REMAINDER인 경우, 데이터를 끝까지 읽음
  - 해당 데이터에 대한 포인터를 반환함

- 파라미터
  - ObjectID \*oid  
(IN) 읽을 object의 ID
  - Four start  
(IN) 읽을 object의 데이터 영역 상에서 읽기를 시작할 offset
  - Four length  
(IN) 읽을 데이터의 길이
  - char \*buf  
(OUT) 읽은 데이터에 대한 포인터
- 반환값
  - Four 실제 읽은 byte 수  
또는 에러코드
- 관련 함수  
EduOM\_ReadObject(), BfM\_GetTrain(), BfM\_FreeTrain()

# EduOM\_NextObject()

- 파일: EduOM\_NextObject.c
- 설명
  - 현재 object의 다음 object의 ID를 반환함
    - 파라미터로 주어진 *curOID*가 NULL 인 경우,
      - File의 첫 번째 object (다음 중 하나) 의 ID를 반환함
        - » 첫 번째 page의 slot array 상에서의 첫 번째 object
        - » 첫 번째 page가 비어있는 경우, 다음 page의 첫 번째 object
      - File이 비어있는 경우, EOS (End Of Scan) 를 반환함
    - 파라미터로 주어진 *curOID*가 NULL 이 아닌 경우,
      - *curOID*에 대응하는 object를 탐색함
      - Slot array 상에서, 탐색한 object의 다음 object의 ID를 반환함
        - » 탐색한 object가 page의 마지막 object인 경우,
          - 다음 page의 첫 번째 object의 ID를 반환함
        - » 탐색한 object가 file의 마지막 page의 마지막 object인 경우,
          - EOS 를 반환함

- 파라미터

- ObjectID \*catObjForFile  
(IN) 현재 object가 저장된 file에 대한 정보 (*sm\_CatOverlayForData*) 가 저장된 object의 ID
- ObjectID \*curOID  
(IN) 현재 object의 ID
- ObjectID \*nextOID  
(OUT) 현재 object의 다음 object의 ID
- ObjectHdr \*objHdr  
(OUT) 현재 object의 다음 object의 header

- 반환값

- Four EOS  
또는 에러코드

- 관련 함수

BfM\_GetTrain(), BfM\_FreeTrain()



# EduOM\_PrevObject()

- 파일: EduOM\_PrevObject.c
- 설명
  - 현재 object의 이전 object의 ID를 반환함
    - 파라미터로 주어진 *curOID*가 NULL 인 경우,
      - File의 마지막 page의 slot array 상에서의 마지막 object의 ID를 반환함
      - File이 비어있는 경우, EOS (End Of Scan) 를 반환함
    - 파라미터로 주어진 *curOID*가 NULL 이 아닌 경우,
      - *curOID*에 대응하는 object를 탐색함
      - Slot array 상에서, 탐색한 object의 이전 object의 ID를 반환함
        - » 탐색한 object가 page의 첫 번째 object인 경우,
          - 이전 page의 마지막 object의 ID를 반환함
          - 이전 page가 비어있는 경우, EOS를 반환함
        - » 탐색한 object가 file의 첫 번째 page의 첫 번째 object인 경우,
          - EOS를 반환함


- 파라미터
  - ObjectID \*catObjForFile  
(IN) 현재 object가 저장된 file에 대한 정보 (*sm\_CatOverlayForData*) 가 저장된 object의 ID
  - ObjectID \*curOID  
(IN) 현재 object의 ID
  - ObjectID \*prevOID  
(OUT) 현재 object의 이전 object의 ID
  - ObjectHdr \*objHdr  
(OUT) 현재 object의 이전 object의 header
  
- 반환값
  - Four EOS  
또는 에러코드
  
- 관련 함수  
BfM\_GetTrain(), BfM\_FreeTrain()

# 구현할 Internal Function

- `eduum_CreateObject()`

# eduum\_CreateObject()

- 파일: eduum\_CreateObject.c
- 설명
  - File을 구성하는 page들 중 파라미터로 지정한 object와 같은 (또는 인접한) page에 새로운 object를 삽입하고, 삽입된 object의 ID를 반환함
    - Object 삽입을 위해 필요한 자유 공간의 크기를 계산함
      - `sizeof(ObjectHdr)` + align 된 object 데이터 영역의 크기 + `sizeof(SlottedPageSlot)`
    - Object를 삽입할 page를 선정함
      - 파라미터로 주어진 *nearObj*가 NULL 이 아닌 경우,
        - » *nearObj*가 저장된 page에 여유 공간이 있는 경우,
          - 해당 page를 object를 삽입할 page로 선정함
          - 선정된 page를 현재 available space list에서 삭제함
          - 필요시 선정된 page를 compact 함

- » *nearObj*가 저장된 page에 여유 공간이 없는 경우,
  - 새로운 page를 할당 받아 object를 삽입할 page로 선정함
  - 선정된 page의 header를 초기화함
  - 선정된 page를 file 구성 page들로 이루어진 list에서 *nearObj*가 저장된 page의 다음 page로 삽입함
- 파라미터로 주어진 *nearObj*가 NULL 인 경우, 
  - » Object의 크기가 page의 data 영역 전체 크기의 절반 이하이고, 해당 object를 삽입할 수 있는 크기의 available space를 갖는 available space list들 중 가장 작은 크기의 available space를 갖는 list에 page가 존재하는 경우,
    - 해당 available space list의 첫 번째 page를 object를 삽입할 page로 선정함
    - 선정된 page를 현재 available space list에서 삭제함
    - 필요시 선정된 page를 compact 함
  - » 위 조건을 만족하지 않고, file의 마지막 page (file을 구성하는 page들의 linked list에서의 마지막 page)에 여유 공간이 있는 경우,
    - File의 마지막 page를 object를 삽입할 page로 선정함
    - 필요시 선정된 page를 compact 함

- » 그 외의 경우,
    - 새로운 page를 할당 받아 object를 삽입할 page로 선정함
    - 선정된 page의 header를 초기화함
    - 선정된 page를 file의 구성 page들로 이루어진 list에서 마지막 page로 삽입함
- 선정된 page에 object를 삽입함
  - Object의 header를 갱신함
    - »  $length :=$  데이터의 길이
  - 선정한 page의 contiguous free area에 object를 복사함
  - Slot array의 빈 slot 또는 새로운 slot 한 개를 할당 받아 복사한 object의 식별을 위한 정보를 저장함
  - Page의 header를 갱신함
    - » Object 삽입으로 인해 사용되는 자유 공간의 크기  
 $= \text{sizeof}(\text{ObjectHdr}) + \text{align 된 object 데이터 영역의 크기}$   
 $+ \text{새로운 slot을 할당 받은 경우의 sizeof}(\text{SlottedPageSlot})$
  - Page를 알맞은 available space list에 삽입함
- 삽입된 object의 ID를 반환함

- 파라미터

- ObjectID \*catObjForFile  
(IN) object를 삽입할 file에 대한 정보 (*sm\_CatOverlayForData*) 가 저장된 object의 ID
- ObjectID \*nearObj  
(IN) 삽입할 object가 인접 저장 되어야 하는 object의 ID
- ObjectHdr \*objHdr  
(IN) 삽입할 object의 tag 값이 저장된 object header
- Four length  
(IN) 삽입할 object의 데이터의 길이 (단위: # of bytes)
- char \*data  
(IN) 삽입할 object의 데이터
- ObjectID \*oid  
(OUT) 삽입된 object의 ID

- 반환값
  - Four 에러코드

- 관련 함수

EduOM\_CompactPage(), om\_GetUnique(), om\_FileMapAddPage(),  
om\_PutInAvailSpaceList(), om\_RemoveFromAvailSpaceList(),  
RDsM\_PageIdToExtNo(), RDsM\_AllocTrains(), BfM\_GetTrain(),  
BfM\_GetNewTrain(), BfM\_FreeTrain(), BfM\_SetDirty()



# 제공되는 API Function 들

- RDsM\_PageIdToExtNo()
  - Page가 속한 extent의 번호를 반환함
    - Extent: 물리적으로 인접한 page들의 집합
  - 파라미터
    - PageID \*pageId  
(IN) page의 ID
    - Four \*extNo  
(OUT) page가 속한 extent의 번호
  - 반환값
    - Four 에러코드

## — 예

```
Four eduom_CreateObject(...)
{
    Four firstExt;    /* number of the first extent of the file */
    sm_CatOverlayForData *catEntry;    /* pointer to the data file catalog information */
    PhysicalFileID pFid;    /* ID of the first page in the file */
    ...
    MAKE_PHYSICALFILEID(pFid, catEntry->fid.volNo, catEntry->firstPage);

    /* Get the first extent number of the file */
    e = RDsM_PageIdToExtNo((PageID *)&pFid, &firstExt);
    if( e < 0 ) ERR( e );
    ...
}
```

- RDsM\_AllocTrains()

- Disk에서 새로운 page (*sizeOfTrain*=1) 또는 train (*sizeOfTrain*>1)을 할당하고, 할당된 page의 ID 또는 train의 첫 번째 page의 ID를 반환함

- Train: page 데이터 영역의 크기보다 큰 large object를 저장하기 위한 구조로서, EduOM에서는 사용하지 않음

- 파라미터

- Four volNo

- (IN) page/train을 할당할 disk가 속한 volume의 번호

- Four firstExt

- (IN) page/train을 할당할 file의 첫 번째 page가 속한 extent의 번호

- PageID \*nearPID

- (IN) 할당할 page/train이 disk 상에서 물리적으로 인접해야 하는 page의 ID

- Two eff

- (IN) page/train을 할당할 file의 extent fill factor

- Four numOfTrains  
(IN) 할당할 page/train 개수
  - Two sizeOfTrain  
(IN) 할당할 train의 크기 (단위: # of pages)  
(※ page를 할당하기 위해서는, *sizeOfTrain*을 1로 설정함)
  - PageID \*trainIDs  
(OUT) 할당된 page의 ID 또는 train의 첫 번째 page의 ID
- 반환값
- Four 에러코드

## — 예

```
Four eduom_CreateObject(...)
{
    PageID pid;    /* ID of a page */
    PageID nearPid; /* ID of the page to which the page to be allocated is to be
                    physically adjacent on the disk */
    Four firstExt; /* number of the first extent of the file */
    sm_CatOverlayForData *catEntry; /* pointer to the data file catalog information */
    ...
    /* Allocate a new page */
    e = RDsM_AllocTrains(catEntry->fid.volNo, firstExt, &nearPid, catEntry->eff, 1,
    PAGESIZE2, &pid);
    if( e < 0 ) ERR( e );
    ...
}
```

- BfM\_GetTrain()

- Page (*sizeOfTrain*=1) 또는 train (*sizeOfTrain*>1) 을 buffer에 fix 하고, 해당 page/train에 대한 포인터를 반환함
  - 모든 transaction들은 page/train을 access하기 전에 해당 page/train을 buffer에 fix 해야 함
  - Disk에서 새롭게 할당된 page/train을 buffer에 fix하고자 한다면, 성능 향상을 위해 BfM\_GetTrain() 대신 BfM\_GetNewTrain()을 호출하는 것이 좋음
    - 해당 page/train은 empty page/train이므로 내용을 읽어오기 위해 disk에 assess할 필요가 없으며, BfM\_GetNewTrain()은 disk에 access하지 않고 해당 empty page/train을 buffer에 fix 함
- 파라미터
  - TrainID \*trainId  
(IN) fix 할 page의 ID 또는 train의 첫 번째 page의 ID
  - Char \*\*retBuf  
(OUT) fix된 page/train에 대한 포인터
  - Four type  
(IN) page/train을 fix 할 buffer의 종류  
(※ page를 fix 하기 위해서는, type을 PAGE\_BUF로 설정함)
- 반환값
  - Four 에러코드

## – 예

```
Four eduom_CreateObject(
    ObjectID   *catObjForFile, /* IN: ID of the object that contains the catalog
    information */
    ...)
{
    SlottedPage *catPage; /* pointer to a buffer */
    ...
    /* Fix the page that contains the catalog object to the buffer */
    e = BfM_GetTrain((TrainID*)catObjForFile, (char**)&catPage, PAGE_BUF);
    if( e < 0 ) ERR( e );
    ...
}
```

- BfM\_GetNewTrain()
  - Disk 상에서 새롭게 할당된 page (*sizeOfTrain*=1) 또는 train (*sizeOfTrain*>1) 을 buffer에 fix 하고, 해당 page/train에 대한 포인터를 반환함
  - 파라미터
    - TrainID    \*trainId  
(IN) fix 할 page의 ID 또는 train의 첫 번째 page의 ID
    - char    \*\*retBuf  
(OUT) fix된 page/train에 대한 포인터
    - Four    type  
(IN) page/train을 fix 할 buffer의 종류  
(※ page를 fix 하기 위해서는, *type*을 PAGE\_BUF로 설정함)
  - 반환값
    - Four    에러코드



## — 예

```
Four eduom_CreateObject(...)
{
    PageID pid;    /* ID of a page */
    SlottedPage *apage;    /* pointer to a buffer */
    ...
    /* Fix the page that has been newly allocated on the disk to the buffer */
    e = BfM_GetNewTrain(&pid, (char **)&apage, PAGE_BUF);
    if( e < 0 ) ERR( e );
    ...
}
```

- BfM\_FreeTrain()

- Page (*sizeOfTrain*=1) 또는 train (*sizeOfTrain*>1) 을 buffer에서 unfix 함
  - 모든 transaction들은 page/train access를 마치고 해당 page/train 을 buffer에서 unfix 해야 함
- 파라미터
  - TrainID \*trainId  
(IN) unfix 할 page의 ID 또는 train의 첫 번째 page의 ID
  - Four type  
(IN) page/train이 저장된 buffer의 종류  
(※ page를 unfix 하기 위해서는, *type*을 PAGE\_BUF로 설정함)
- 반환값
  - Four 에러코드

## – 예

```
Four eduom_CreateObject(  
    ObjectID    *catObjForFile, /* IN: ID of the object that contains the catalog  
    information */  
    ...)  
{  
    ...  
    /* Unfix the page that contains the catalog object from the buffer */  
    e = BfM_FreeTrain((TrainID*)catObjForFile, PAGE_BUF);  
    if( e < 0 ) ERR( e );  
    ...  
}
```

- BfM\_SetDirty()
  - Buffer에 저장된 page (*sizeOfTrain*=1) 또는 train (*sizeOfTrain*>1) 이 수정되었음을 표시하기 위해 DIRTY bit를 set 함
  - 파라미터
    - TrainID \*trainId  
(IN) DIRTY bit를 set 할 page의 ID 또는 train의 첫 번째 page의 ID
    - Four type  
(IN) page/train이 저장된 buffer의 종류  
(※ page의 DIRTY bit를 set 하기 위해서는, *type*을 PAGE\_BUF로 설정함)
  - 반환값
    - Four 에러코드

— 예

```
Four eduom_CreateObject(...)
{
    PageID pid;    /* ID of a page */
    ...
    /* Set the DIRTY bit */
    e = BfM_SetDirty(&pid, PAGE_BUF);
    if (e < 0) ERRB1(e, &pid, PAGE_BUF);
    ...
}
```

- Util\_getElementFromPool()
  - Pool에서 새로운 dealloc list element 한 개를 위한 메모리 공간을 할당 받고, 할당 받은 메모리 공간에 대한 포인터를 반환함
  - 파라미터
    - Pool \*aPool  
(IN) 할당을 위해 사용할 element pool
    - void \*elem  
(OUT) 할당 받은 dealloc list element
  - 반환값
    - Four 예러코드

## — 예

```
Four EduOM_DestroyObject(...
    Pool    *dlPool,      /* INOUT: pool of the elements of the dealloc list */
    DeallocListElem *dlHead) /* INOUT: head of the dealloc list */
{
    PageID pid; /* ID of a page */
    DeallocListElem *dlElem; /* pointer to the element of the dealloc list */
    ...
    /* Insert the deallocated page into the dealloc list */
    e = Util_getElementFromPool(dlPool, &dlElem);
    if( e < 0 ) ERR( e );

    dlElem->type = DL_PAGE;
    dlElem->elem.pid = pid; /* ID of the deallocated page */
    dlElem->next = dlHead->next;
    dlHead->next = dlElem;
    ...
}
```

# 제공되는 Function 들

- om\_GetUnique()
  - Page에서 사용할 unique 번호를 할당 받고, 해당 page의 header의 관련 정보를 갱신하고, 할당 받은 unique 번호를 반환함
  - 파라미터
    - PgaeID \*pid  
(IN) unique 번호를 할당 받을 page의 ID
    - Unique \*unique  
(OUT) 할당 받은 unique 번호
  - 반환값
    - Four 에러코드



## — 예

```
Four eduom_CreateObject(...)
{
    PageID pid;    /* ID of a page */
    SlottedPage *apage;    /* pointer to a buffer */
    ...
    /* Allocate a unique number to be used for the page */
    e = om_GetUnique(&pid, &(apage->slot[-i].unique));
    if (e < 0) ERRB1(e, &pid, PAGE_BUF);
    ...
}
```

- om\_FileMapAddPage()
  - Page를 file 구성 page들로 이루어진 list에 삽입함
  - 파라미터
    - ObjectID \*catObjForFile  
(IN) file에 대한 정보가 저장된 object의 ID
    - PageID \*prevPID  
(IN) 삽입할 page의 이전 page로 설정될 page의 ID
    - PageID \*newPID  
(IN) 삽입할 page의 ID
  - 반환값
    - Four 에러코드

## — 예

```
Four eduom_CreateObject(  
    ObjectID *catObjForFile, /* IN: ID of the object that contains the catalog  
information */  
    ObjectID *nearObj, /* IN: create the new object near this object */  
    ...)  
{  
    PageID pid; /* ID of a page */  
    ...  
    /* Insert the page into the list of pages of the file */  
    e = om_FileMapAddPage(catObjForFile, (PageID *)nearObj, &pid);  
    if (e < 0) ERRB1(e, &pid, PAGE_BUF);  
    ...  
}
```

- om\_FileMapDeletePage()
  - Page를 file 구성 page들로 이루어진 list에서 삭제함
  - 파라미터
    - ObjectID \*catObjForFile  
(IN) file에 대한 정보가 저장된 object의 ID
    - PageID \*newPID  
(IN) 삭제할 page의 ID
  - 반환값
    - Four 에러코드

## — 예

```
Four EduOM_DestroyObject(
    ObjectID   *catObjForFile, /* IN: ID of the object that contains the catalog
    information */
    ...)
{
    PageID pid;   /* ID of a page */
    ...
    /* Delete the page from the list of pages of the file */
    e = om_FileMapDeletePage(catObjForFile, &pid);
    if (e < 0) ERRB1(e, &pid, PAGE_BUF);
    ...
}
```

- om\_PutInAvailSpaceList()
  - Page를 available space list에 삽입함
  - 파라미터
    - ObjectID \*catObjForFile  
(IN) file에 대한 정보가 저장된 object의 ID
    - PageID \*pid  
(IN) 삽입할 page의 ID
    - SlottedPage \*apage  
(INOUT) 삽입할 page
  - 반환값
    - Four 에러코드

## — 예

```
Four eduom_CreateObject(
    ObjectID *catObjForFile, /* IN: ID of the object that contains the catalog
information */
    ...)
{
    PageID pid; /* ID of a page */
    SlottedPage *apage; /* pointer to a buffer */
    ...
    /* Insert the page into the available space list */
    e = om_PutInAvailSpaceList(catObjForFile, &pid, apage);
    if (e < 0) ERRB1(e, &pid, PAGE_BUF);
    ...
}
```

- om\_RemoveFromAvailSpaceList()
  - Page를 available space list에서 삭제함
  - 파라미터
    - ObjectID \*catObjForFile  
(IN) file에 대한 정보가 저장된 object의 ID
    - PageID \*pid  
(IN) 삭제할 page의 ID
    - SlottedPage \*apage  
(INOUT) 삭제할 page
  - 반환값
    - Four 에러코드



## — 예

```
Four eduom_CreateObject(
    ObjectID *catObjForFile, /* IN: ID of the object that contains the catalog
information */
    ...)
{
    PageID pid; /* ID of a page */
    SlottedPage *apage; /* pointer to a buffer */
    ...
    /* Delete the page from the available space list */
    e = om_RemoveFromAvailSpaceList(catObjForFile, &pid, apage);
    if (e < 0) ERRB1(e, &pid, PAGE_BUF);
    ...
}
```

# Error 처리

- Error 처리 매크로
  - ERR(e)
    - 파라미터로 주어진 error code *e*, error가 발생한 파일명 및 error가 발생한 위치 등을 error log 파일 (odysseus\_error.log) 에 기록한 후, error code를 반환함
    - 사용예  
if(length < 0) ERR(eBADLENGTH\_OM)
  - ERRB1(e, pid, t)
    - Error code *e*를 반환하기 전에 파라미터로 주어진 *pid*에 대응하는 page를 unfix 하는 것을 제외하고 ERR(e) 와 동일함
    - 사용예  
if(e < 0) ERRB1(e, &pid, PAGE\_BUF)
- Error code  
\$(EduOM\_HOME\_DIR)/Header/EduOM\_errorcodes.h 파일 참고

# Project 수행 방법

- Project에서 사용되는 파일
  - 학생들이 구현해야 하는 파일
    - Skeleton 파일 (.c 파일)  
구현부가 생략되어 있는 function들로 구성된 파일
  - 학생들에게 주어지는 파일
    - Object 파일 (.o 파일)  
기반 시스템인 오디세우스/COSMOS가 object 파일로 compile된 것으로서, 구현할 모듈에서 사용되는 하위 레벨 function 들을 포함한 오디세우스/COSMOS의 모든 function들이 포함된 파일
    - Header 파일 (.h 파일)  
구현할 모듈 및 테스트 모듈과 관련된 데이터 구조 정의와 function들의 prototype 들로 구성된 파일
    - 테스트 모듈 소스 코드 파일  
구현한 모듈의 기능을 테스트 하기 위한 테스트 모듈의 소스 코드 파일
    - Solution 실행 파일  
정확한 테스트 결과를 보여주는 실행 파일

- Project 수행 방법

- Skeleton 파일 내의 function들을 구현함
  - 구현시 \$(EduOM\_HOME\_DIR)/Header 디렉토리의 헤더 파일들에 저장된 각종 macro들을 활용 가능함
- make clean (object 파일 및 생성된 DB 볼륨 삭제), make (컴파일) 명령을 이용하여, 구현된 skeleton 파일들과 테스트 모듈 소스 코드 파일을 compile하고 주어진 object 파일과 link함
  - Compile 및 linking 결과로서 구현된 모듈의 기능을 테스트하기 위한 실행 파일이 생성됨
- 생성된 실행 파일의 실행 결과를 주어진 solution 실행 파일의 실행 결과와 비교함

- ❖ 일부 기능만을 구현하여 테스트 하는 방법

- ❖ \$(EduOM\_HOME\_DIR)/Header/EduOM\_TestModule.h 파일 코드를 변경
  - ❖ 구현한 API 함수의 경우, 대응하는 매크로 값을 TRUE로 변경
  - ❖ 구현하지 않은 API 함수의 경우, 대응하는 매크로 값을 FALSE로 변경
- ❖ Make 명령어를 입력하여 project를 recompile 함

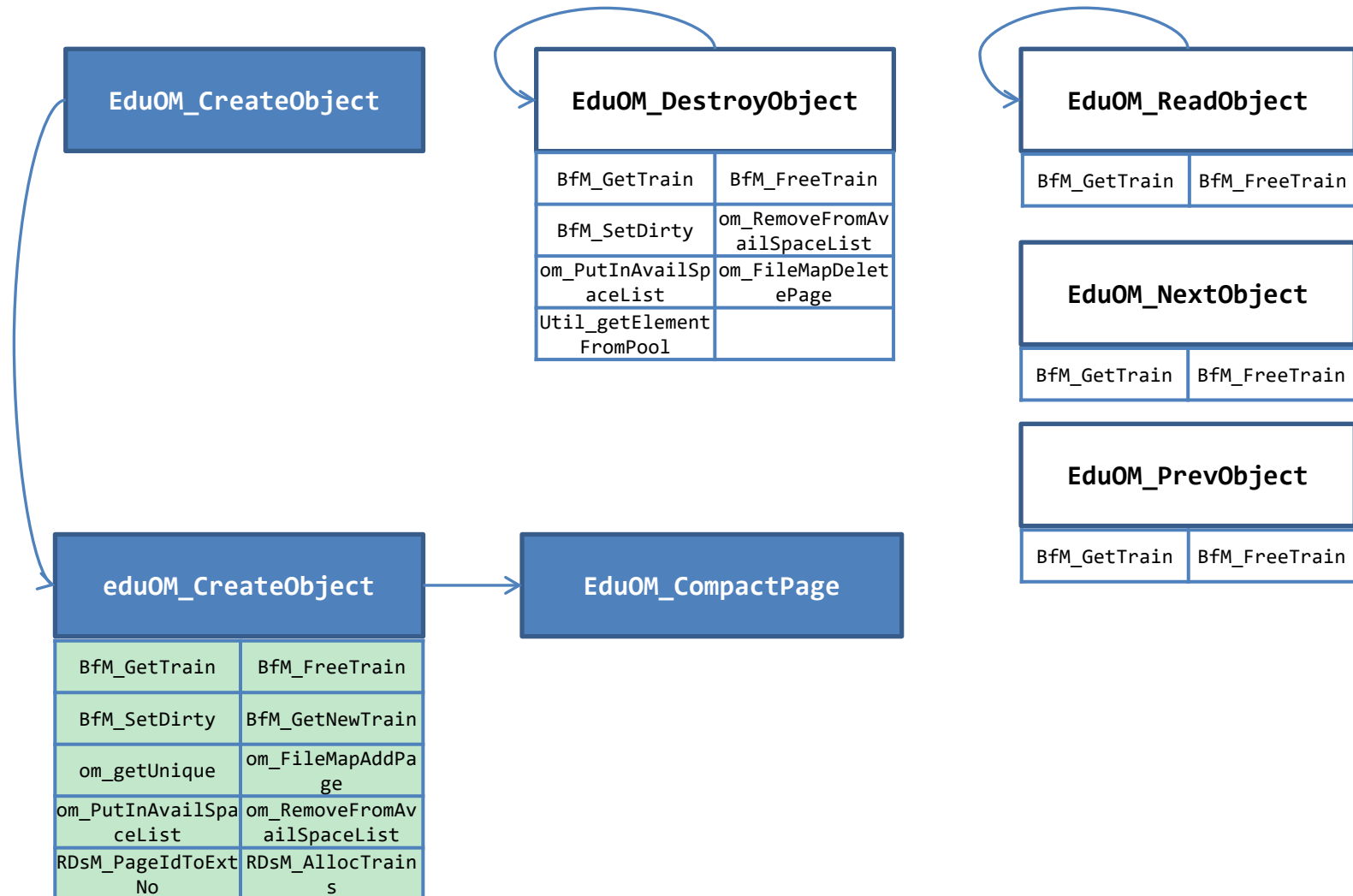
- ❖ API function에서 호출하는 일부 internal function을 구현하지 않고 해당 API function을 구현하는 방법

- ❖ 대응하는 default solution function (internal function 명에서 "edu" keyword가 제거된 형태)을 사용함
  - ❖ 예: eduom\_CreateObject()의 successful default solution function은 om\_CreateObject() 임

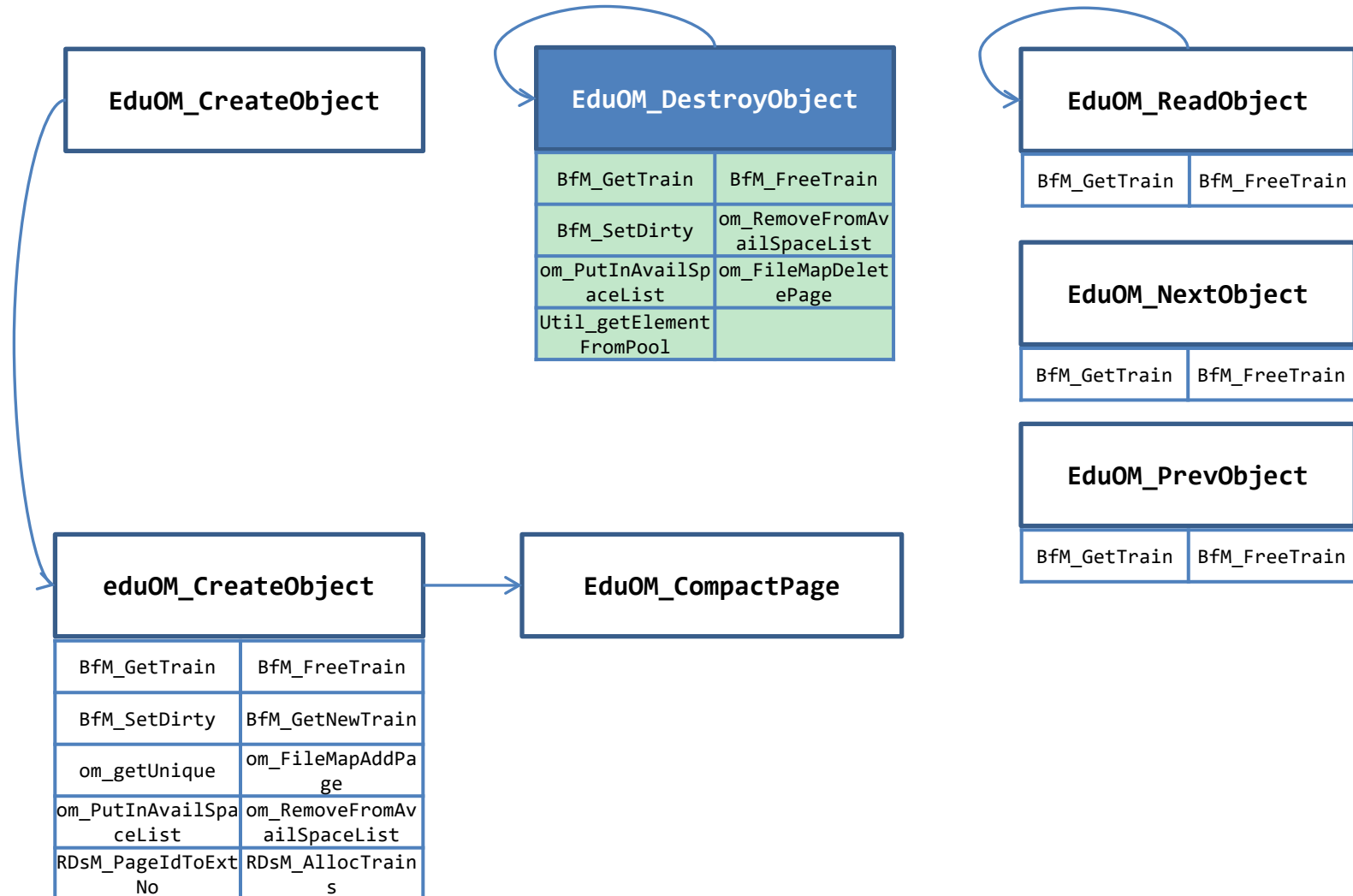
# Appendix

Function Call Graph

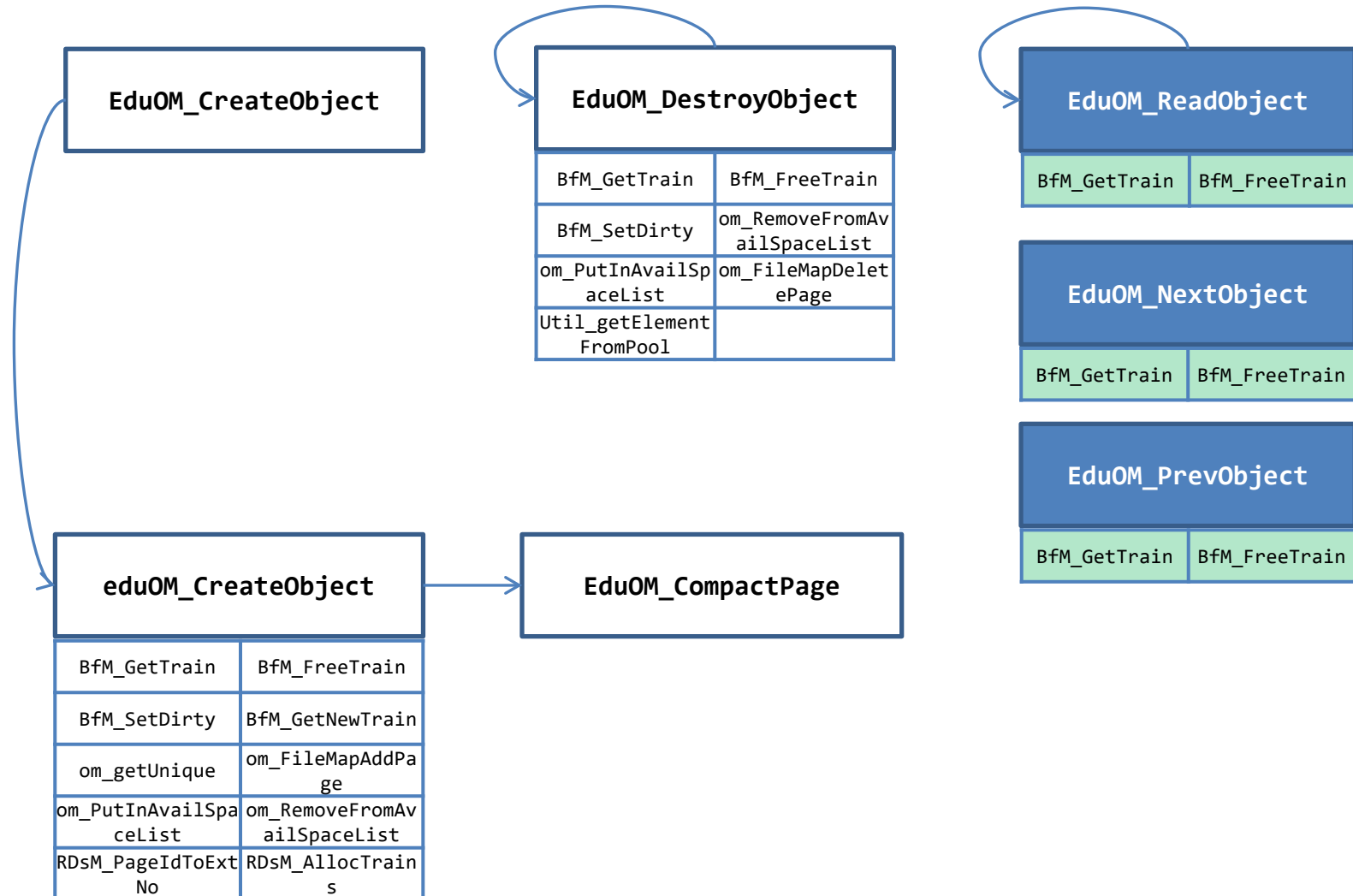
# EduOM\_CreateObject



# EduOM\_DestroyObject



# EduOM\_ReadObject, EduOM\_NextObject, EduOM\_PrevObject





# EduOM\_CompactPage

