

# 오디세우스/EduCOSMOS Project #3: EduBtM Project Manual

Version 1.1

September 2013

Instructor: Prof. Kyu-Young Whang

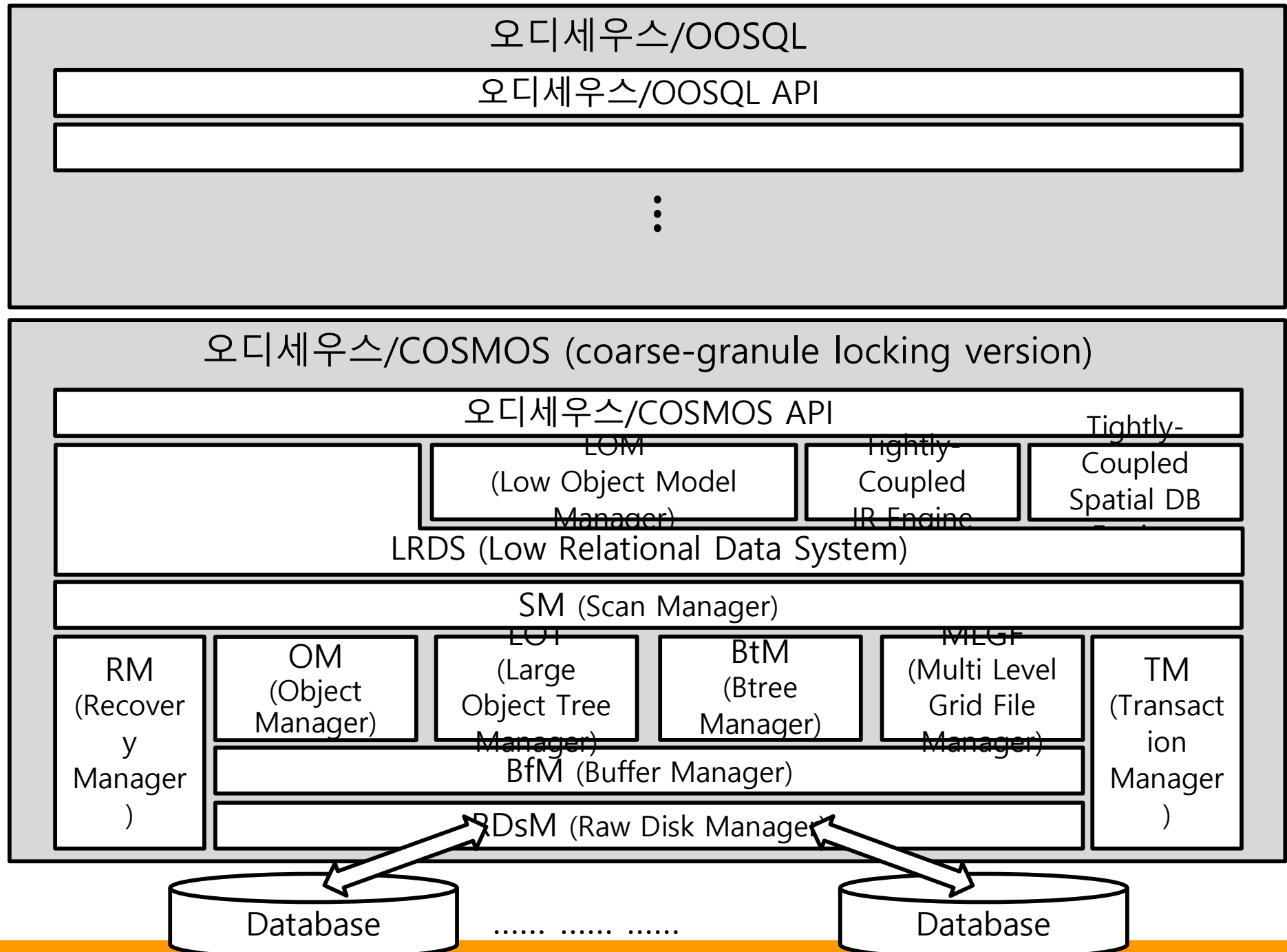
# 목차

- 소개
  - 오디세우스/COSMOS
  - 오디세우스/EduCOSMOS Project
- EduBtM Project
  - 자료 구조 및 연산
  - 구현할 Function 들
  - 제공되는 Function 들
  - Error 처리
- Project 수행 방법

# 오디세우스/COSMOS

- 오디세우스
  - 1990년부터 한국과학기술원 데이터베이스 및 멀티미디어 연구실에서 개발한 객체 관계형 DBMS
- 오디세우스/COSMOS
  - 오디세우스의 저장 시스템으로서, 각종 데이터베이스 응용 소프트웨어의 하부 구조로 사용되고 있음

# 오디세우스 구조



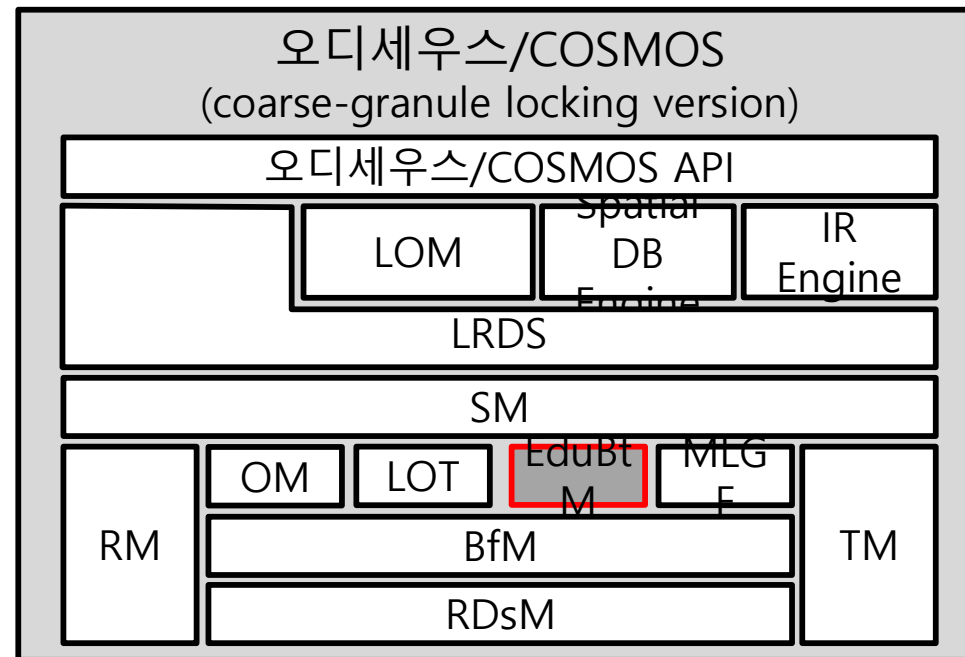
# 오디세우스/EduCOSMOS Project

- 개요
  - Coarse granule locking 버전 오디세우스/COSMOS의 일부분을 구현하는 교육 목적용 project
- 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

# EduBtM Project

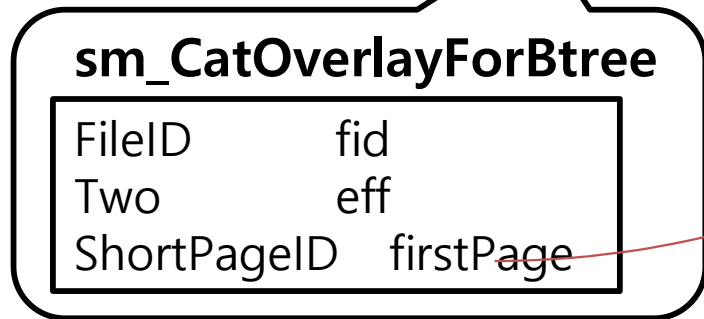
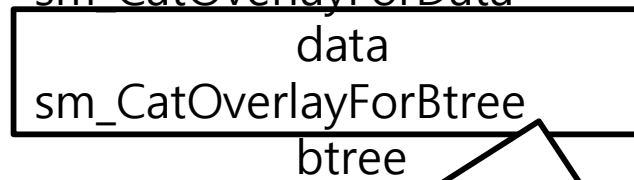
- 목표

- B+ tree 색인 및 색인 page 관련 구조에 대한 연산들을 구현함
- EduBtM에서는 오디세우스/COSMOS BtM의 기능들 중 극히 제한된 일부 기능들만을 구현함

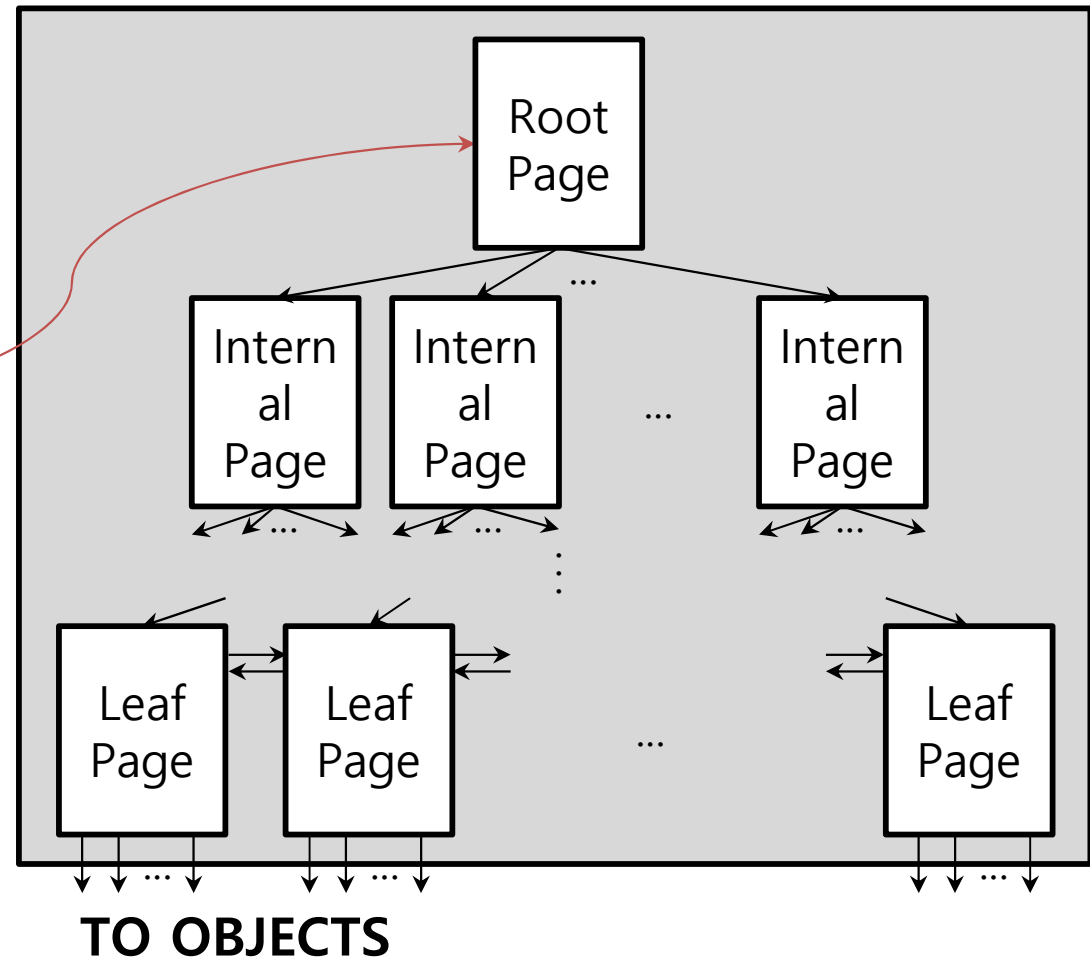


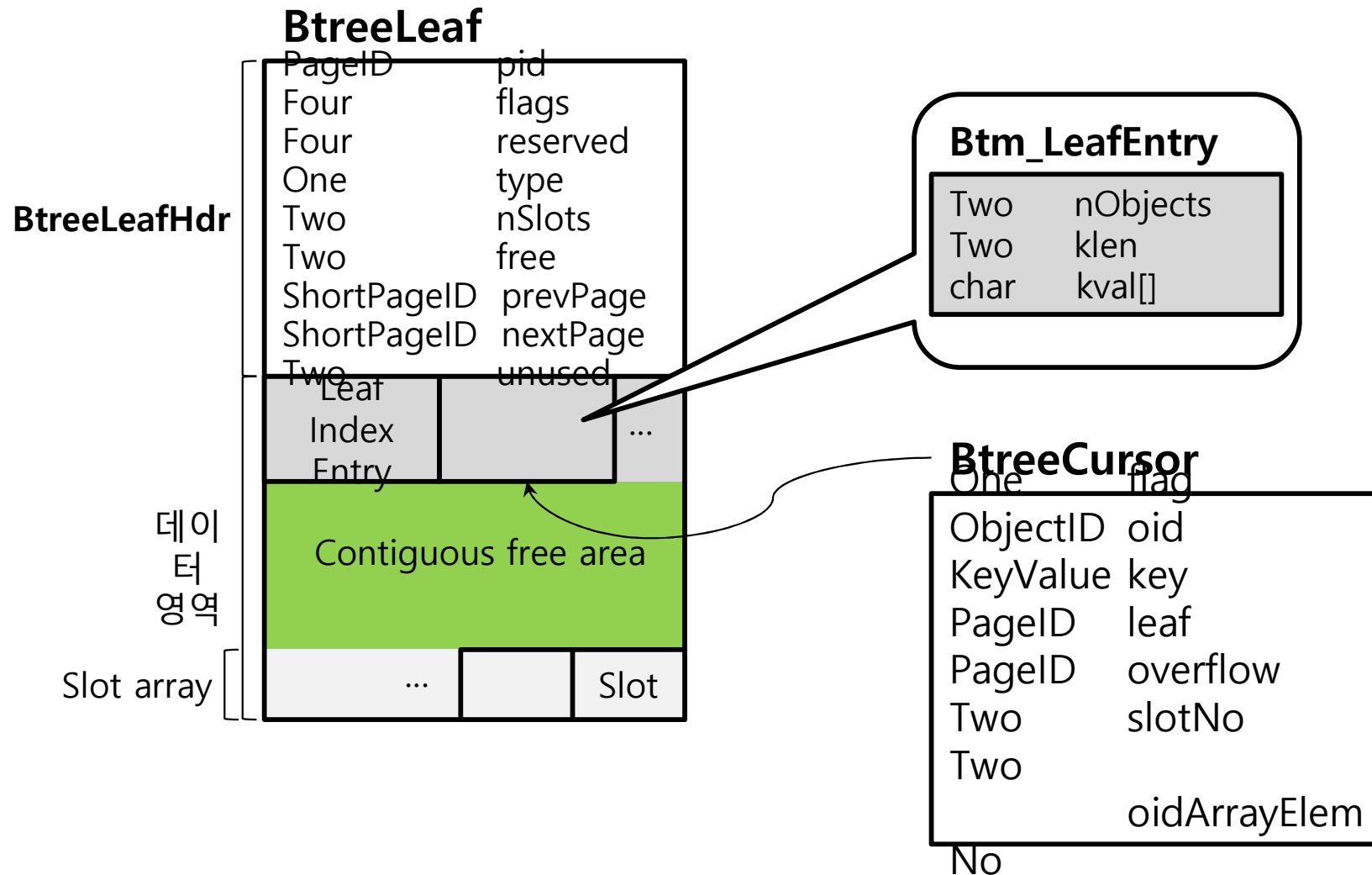
# 데이터 구조

## sm\_CatOverlayForSysTables

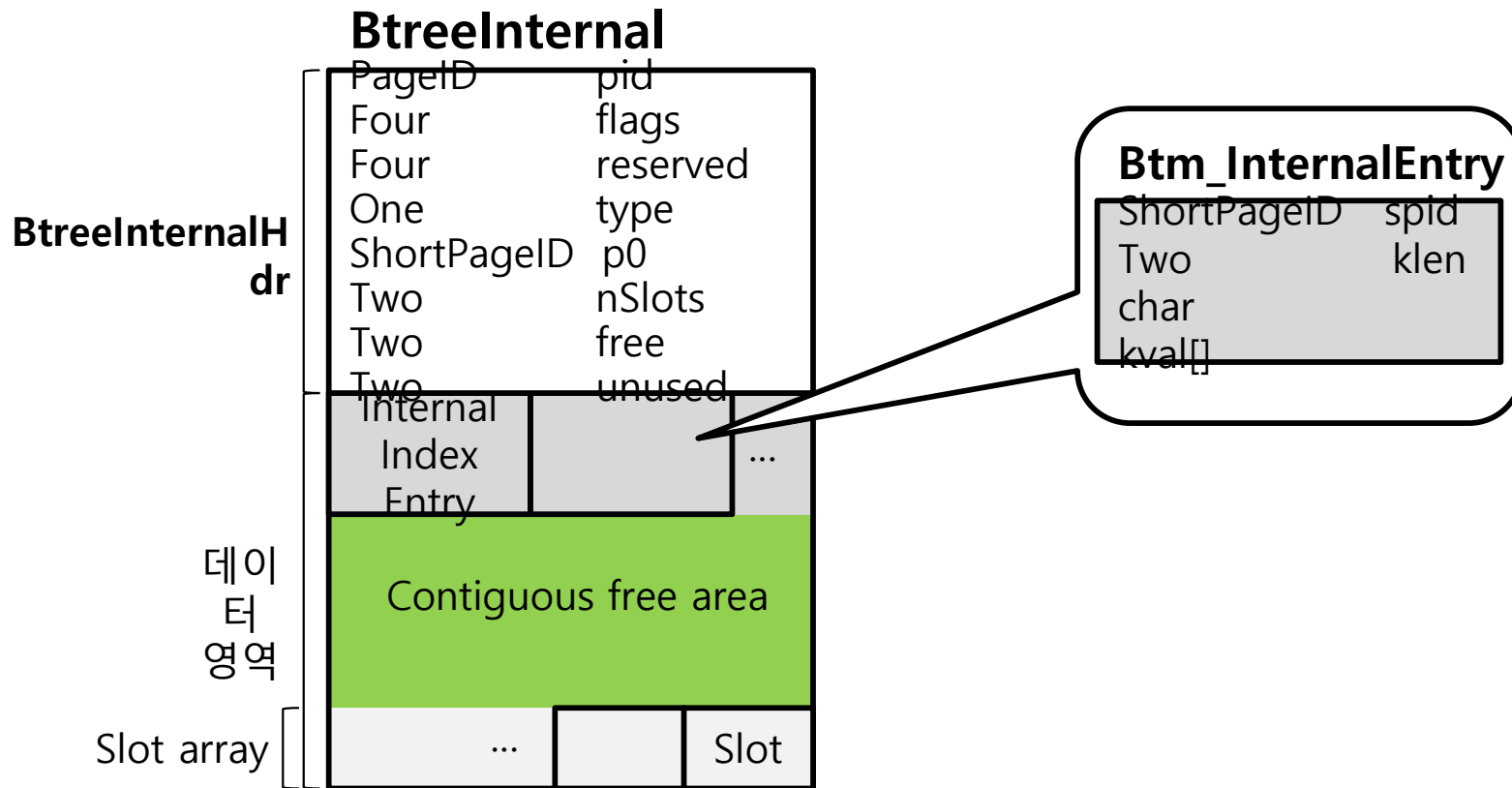


## B+ Tree Index









# sm\_CatOverlayForSysTables

- 개요

- 데이터 file 및 관련 색인 file에 대한 정보를 저장하기 위한 데이터 구조
  - 데이터 file: 서로 관련 있는 object들이 저장된 page들의 집합
  - 색인 file: 데이터 file에 생성된 B+ tree 색인들을 구성하는 page들의 집합

- 구성

- data
  - 데이터 file에 대한 정보를 저장하기 위한 데이터 구조
    - EduOM project manual 참고
- btree
  - 색인 file에 대한 정보를 저장하기 위한 데이터 구조

# sm\_CatOverlayForBtree

- 개요
  - 색인 file에 대한 정보를 저장하기 위한 데이터 구조
- 구성
  - fid
    - 색인 file의 ID
  - eff
    - 색인 file의 extent fill factor로서, EduBtM에서는 사용하지 않음
  - firstPage
    - 색인 file을 구성하는 첫 번째 page의 번호
      - 최초 생성된 B+ tree 색인의 root page의 번호

# BtreeLeaf

- 개요
  - B+ tree 색인의 leaf node를 나타내는 page 데이터 구조
- 구성
  - hdr
    - Page에 대한 정보를 저장하는 page header
  - data[]
    - Leaf index entry들을 저장하는 데이터 영역
      - Index entry로서 <object의 key, object ID> pair 및 관련 정보가 저장됨
      - 데이터 영역은 항상 50% 이상 채워져 있어야 함
        - » 예외: page가 root인 경우

## – slot[1]

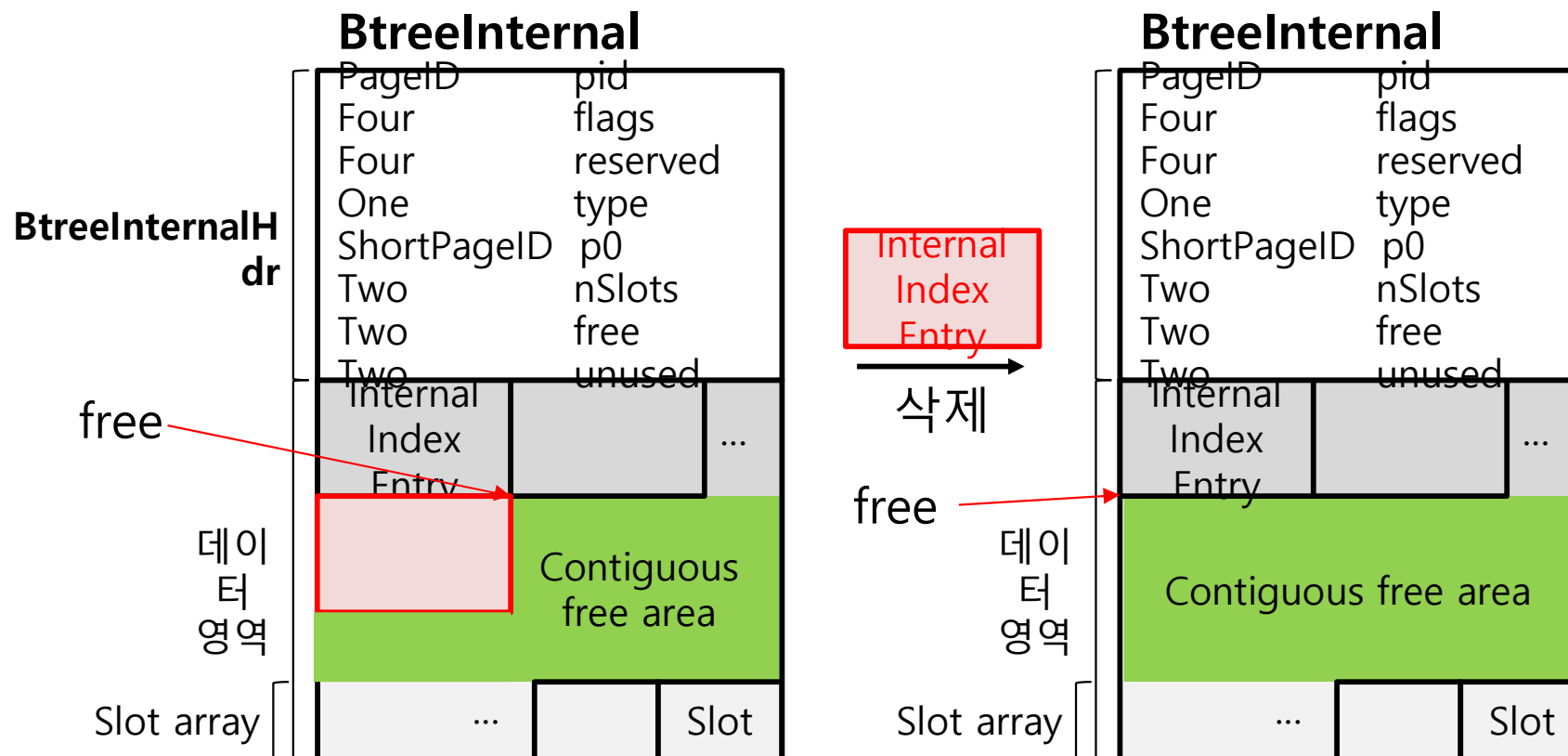
- Page의 데이터 영역에 저장된 index entry의 offset을 저장하는 slot들의 array
  - 첫 번째 slot ( $slot[0]$ ) 을 제외한 나머지 slot들 ( $slot[1] \sim slot[n]$ ) 은 메모리 공간을 데이터 영역과 공유함
    - » 첫 번째 slot의 array index = 0
    - » 다음 slot의 array index = 이전 slot의 array index – 1
    - » Slot 번호 = |slot의 array index|
  - 효율적인 검색을 위해 index entry의 offset들은 index entry의 key 순으로 정렬되어 slot들에 저장됨
    - » 첫 번째 slot: page 내에서 가장 작은 key 값을 갖는 index entry의 offset이 저장됨
    - » 마지막 slot: page 내에서 가장 큰 key 값을 갖는 index entry의 offset이 저장됨
    - » 데이터 영역상에서의 index entry들 자체의 정렬은 필요 없음

# BtreeLeafHdr

- 개요
  - B+ tree 색인의 leaf page에 대한 정보를 저장하기 위한 데이터 구조
- 구성
  - pid
    - Page의 ID
  - flags
    - Page의 type을 나타내는 bit들의 집합
      - 첫 번째 bit 및 세 번째 bit가 모두 set 된 경우 (BTREE\_PAGE\_TYPE): B+ tree 색인 page임을 나타냄
      - 이외의 bit들은 EduBtM에서는 사용하지 않음
  - reserved
    - Page에 대한 추가적인 정보를 저장하기 위한 예비 변수
  - type
    - B+ tree 색인 page의 type을 나타내는 bit들의 집합
      - 첫 번째 bit가 set 된 경우 (ROOT): root page임을 나타냄
      - 두 번째 bit가 set 된 경우 (INTERNAL): internal page임을 나타냄

- 세 번째 bit가 set 된 경우 (LEAF): leaf page임을 나타냄
  - 두 번째 bit 및 네 번째 bit가 모두 set 된 경우 (FREEPAGE): deallocate 될 page임을 나타냄
  - 이외의 bit들은 EduBtM에서는 사용하지 않음
- nSlots
  - Page의 slot array의 크기  
(= 사용중인 slot들 중 마지막 slot의 번호 + 1)
    - Index entry가 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 index entry 이후의 연속된 자유 영역. (※ 단, 마지막 index entry가 삭제되었을 때는 삭제된 index entry의 시작 offset 이후의 연속된 자유 영역이 contiguous free area가 됨.)
- prevPage / nextPage
  - 다음/이전 leaf page의 번호
    - B+ tree 색인의 leaf page들간의 doubly linked list 구조 유지를 위해 사용됨
- unused
  - Page의 데이터 영역 중 contiguous free area를 제외한 자유 영역들의 크기의 합 (단위: bytes)

- 예: 아래 그림에서 contiguous free area와 인접한 빨간색 internal index entry가 삭제될 경우, contiguous free area의 시작 offset인 free 값은 해당 internal index entry의 크기만큼 감소하는 반면, Page의 데이터 영역 중 contiguous free area를 제외한 자유 영역들의 크기의 합인 unused 값은 변동 없음





- prevPage / nextPage
  - 다음/이전 leaf page의 번호
    - B+ tree 색인의 leaf page들간의 doubly linked list 구조 유지를 위해 사용됨
- unused
  - Page의 데이터 영역 중 contiguous free area를 제외한 자유 영역들의 크기의 합 (단위: bytes)

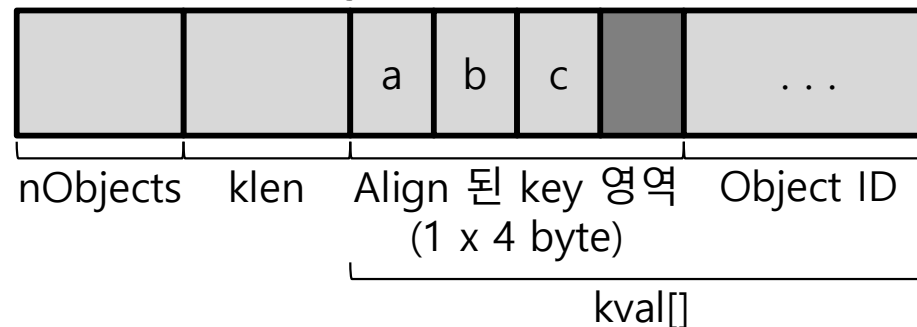
# Btm\_LeafEntry

- 개요
  - B+ tree 색인의 leaf page에 저장되는 leaf index entry를 나타내는 데이터 구조
- 구성
  - nObjects
    - Index entry에 저장된 같은 key 값을 갖는 object의 개수
      - 유일 key를 사용하는 EduBtM에서는 각 key 값 갖는 object는 한 개씩만 존재함
  - klen
    - Index entry에 저장된 key 값의 길이 (단위: bytes)
      - Align 된 key 영역의 크기가 아닌 key 값의 실제 길이

## – kval[]

- <object의 key, object ID> pair가 저장되는 데이터 영역
  - Object의 key 값이 저장된 영역은 4의 배수 (32비트 운영 체제에서의 메모리 할당 기본 단위) 가 되도록 align 됨
    - » 예) 길이가 3인 key 값 "abc" 가 저장된 경우,

### Btm\_LeafEntry



# BtreeInternal

- 개요
  - B+ tree 색인의 internal node를 나타내는 page 데이터 구조
- 구성
  - hdr
    - Page에 대한 정보를 저장하는 page header
  - data[]
    - Internal index entry들을 저장하는 데이터 영역
      - Index entry로서 <object의 key, 자식 page의 번호> pair 및 관련 정보가 저장됨
      - 데이터 영역은 항상 50% 이상 채워져 있어야 함
        - » 예외: page가 root인 경우

## – slot[1]

- Page의 데이터 영역에 저장된 index entry의 offset을 저장하는 slot들의 array
  - 첫 번째 slot ( $slot[0]$ ) 을 제외한 나머지 slot들 ( $slot[1] \sim slot[n]$ ) 은 메모리 공간을 데이터 영역과 공유함
    - » 첫 번째 slot의 array index = 0
    - » 다음 slot의 array index = 이전 slot의 array index + 1
    - » Slot 번호 = |slot의 array index|
  - 효율적인 검색을 위해 index entry의 offset들은 index entry의 key 순으로 정렬되어 slot들에 저장됨
    - » 첫 번째 slot: page 내에서 가장 작은 key 값을 갖는 index entry의 offset이 저장됨
    - » 마지막 slot: page 내에서 가장 큰 key 값을 갖는 index entry의 offset이 저장됨
    - » 데이터 영역상에서의 index entry들 자체의 정렬은 필요 없음

# BtreeInternalHdr

- 개요
  - B+ tree 색인의 internal page에 대한 정보를 저장하기 위한 데이터 구조
- 구성
  - pid
    - Page의 ID
  - flags
    - Page의 type을 나타내는 bit들의 집합
      - 첫 번째 bit 및 세 번째 bit가 모두 set 된 경우 (BTREE\_PAGE\_TYPE): B+ tree 색인 page임을 나타냄
      - 이외의 bit들은 EduBtM에서는 사용하지 않음
  - reserved
    - Page에 대한 추가적인 정보를 저장하기 위한 예비 변수
  - type
    - B+ tree 색인 page의 type을 나타내는 bit들의 집합
      - 첫 번째 bit가 set 된 경우 (ROOT): root page임을 나타냄
      - 두 번째 bit가 set 된 경우 (INTERNAL): internal page임을 나타냄

- 세 번째 bit가 set 된 경우 (LEAF): leaf page임을 나타냄
  - 두 번째 bit 및 네 번째 bit가 모두 set 된 경우 (FREEPAGE): deallocate 될 page임을 나타냄
  - 이외의 bit들은 EduBtM에서는 사용하지 않음
- p0
  - Page의 첫 번째 index entry (slot 번호 = 0) 의 key 값보다 작은 key 값을 갖는 index entry들이 저장된 자식 page의 번호
- nSlots
  - Page의 slot array의 크기  
(= 사용중인 slot들 중 마지막 slot의 번호 + 1)
    - Page의 데이터 영역을 효율적으로 사용하기 위해, index entry가 삽입/삭제 됨에 따라 slot array의 크기가 동적으로 변화함
- free
  - Page의 데이터 영역 중 contiguous free area의 시작 offset
- unused
  - Page의 데이터 영역 중 contiguous free area를 제외한 자유 영역들의 크기의 합 (단위: bytes)

# Btm\_InternalEntry

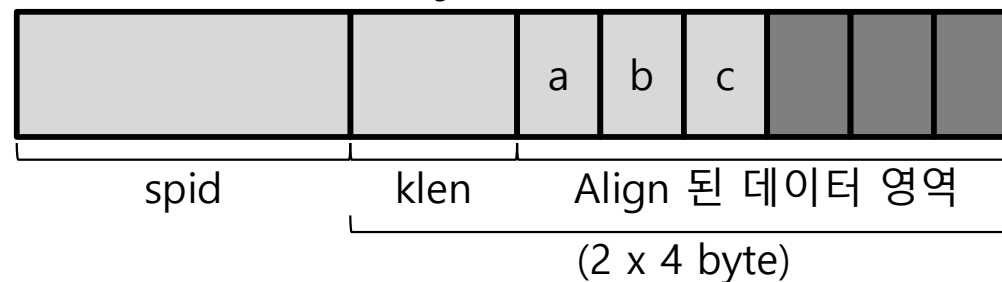
- 개요
  - B+ tree 색인의 internal page에 저장되는 internal index entry를 나타내는 데이터 구조
- 구성
  - spid
    - 현재 index entry의 key 값보다 크거나 같고 다음 index entry의 key 값보다 작은 key 값을 갖는 index entry들이 저장된 자식 page의 번호
  - klen
    - Index entry에 저장된 key 값의 길이 (단위: bytes)
      - Align 된 데이터 영역의 크기가 아닌 key 값의 실제 길이



## – kval[]

- Discriminator key가 저장되는 데이터 영역
  - Discriminator key 값과 *klen*이 저장된 영역은 4 (32비트 운영체제에서의 메모리 할당 기본 단위) 의 배수가 되도록 align 됨
    - » 예) 길이가 3인 key 값 "abc" 가 저장된 경우,

### Btm\_InternalEntry



# KeyValue

- 개요

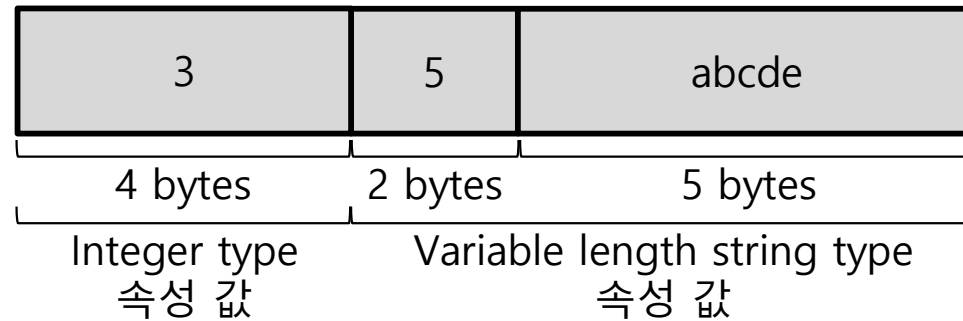
- B+ tree 색인에서 사용되는 key 값 및 관련 정보를 저장하기 위한 데이터 구조
- 다중 속성 key 및 variable length key를 저장 가능함
  - 다중 속성 key: 두 개 이상의 속성들로 구성된 key

- 구성

- len
  - Key 값의 길이
- val[]
  - Key를 구성하는 속성 값들의 sequence
    - Sequence 상에서 각 속성 값을 서로 구분하기 위한 정보는 KeyDesc 데이터 구조에 저장됨

- 속성 값이 variable length string type인 경우, 속성 값이 그 길이와 함께 저장됨
  - » 예) 3 (integer type의 속성 값) 과 "abcde" (variable length string type의 속성 값) 로 구성된 key 값

**val[]**



# KeyDesc

- 개요
  - Key를 구성하는 속성 값들의 sequence에서 각 속성 값을 서로 구분하기 위한 정보를 저장하기 위한 데이터 구조
- 구성
  - flag
    - Key의 type을 나타내는 bit들의 집합
      - 첫 번째 bit가 set 된 경우 (KEYFLAG\_UNIQUE): 유일 key임을 나타냄
      - 이외의 bit들은 EduBtM에서는 사용하지 않음
  - nparts
    - Key를 구성하는 속성 값들의 수

## – kpart[]

- Key의 각 속성에 대한 정보를 저장하는 array 데이터 구조

- type

- » 속성의 type

- Integer type인 경우, type := SM\_INT
    - Variable length string type인 경우, type := SM\_VARSTRING
    - 이외의 type들은 EduBtM에서는 사용하지 않음

- offset

- » *KeyValue*의 *val[]* array에 저장된 속성 값의 offset으로서, EduBtM에서는 사용하지 않음

- length

- » 속성 값의 길이

- » Variable length string type의 속성 값의 경우 사용하지 않음

# BtreeCursor

- 개요
  - B+ tree 색인의 leaf index entry를 가리키는 cursor를 나타내는 데이터 구조
  - 검색 조건을 만족하는 object들에 대한 순차 탐색시 현재 탐색중인 leaf index entry 정보 및 다음 탐색할 leaf index entry 정보를 저장하기 위해 사용됨
- 구성
  - flag
    - Cursor의 상태를 나타내는 변수
      - CURSOR\_ON: cursor가 검색 조건을 만족하는 leaf index entry를 가리키고 있음을 나타냄
      - CURSOR\_EOS: 검색 조건을 만족하는 leaf index entry가 없음을 나타냄 (탐색 종료 의미함)
  - oid
    - Cursor가 가리키는 leaf index entry에 저장된 object ID (OID)

- key
  - Cursor가 가리키는 leaf index entry의 key 값
- leaf
  - Cursor가 가리키는 leaf index entry가 저장된 leaf page의 page ID
- overflow
  - 중복 key 사용시 동일 key 값을 갖는 object들의 ID (OID) 들이 저장된 page의 page ID로서, 유일 key만을 사용하는 EduBtM에서는 사용하지 않음
- slotNo
  - Cursor가 가리키는 leaf index entry의 slot 번호
- oidArrayElmNo
  - 중복 key 사용시 동일 key 값을 갖는 object들의 ID (OID) 들이 저장된 array의 index로서, 유일 key만을 사용하는 EduBtM에서는 사용하지 않음

# 관련 연산

- B+ tree 색인을 생성 / 삭제
- B+ tree 색인에 새로운 object를 삽입
  - 삽입으로 인한 overflow 발생시 split 연산을 고려함
- B+ tree 색인에서 object를 삭제
  - 삭제로 인한 underflow 발생시 merge/redistribute 연산을 제공되는 function들을 통해 처리함
- B+ tree 색인에서 object를 검색



# 구현할 API Function 들

- EduBtM\_CreateIndex()
- EduBtM\_DropIndex()
- EduBtM\_InsertObject()
- EduBtM\_DeleteObject()
- EduBtM\_Fetch()
- EduBtM\_FetchNext()

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

(※ API: Application Programming Interface)

# EduBtM\_CreateIndex()

- 파일: EduBtM\_CreateIndex.c
- 설명
  - 색인 file에서 새로운 B+ tree 색인을 생성하고, 생성된 색인의 root page의 page ID를 반환함
    - btm\_AllocPage()를 호출하여 색인 file의 첫 번째 page를 할당 받음
      - 첫 번째 page의 번호는 *sm\_CatOverlayForBtree*의 *firstPage* 변수에 저장되어 있음
    - 할당 받은 page를 root page로 초기화함
    - 초기화된 root page의 page ID를 반환함

- 파라미터
  - ObjectID \*catObjForFile  
(IN) B+ tree 색인을 생성할 색인 file 및 색인될 데이터 file에 대한 정보 (sm\_CatOverlayForSysTables) 가 저장된 object의 OID
  - PageID \*rootPid  
(OUT) 생성된 B+ tree 색인의 root page의 page ID
- 반환값
  - Four 에러코드
- 관련 함수  
edubtm\_InitLeaf(), btm\_AllocPage(), BfM\_GetTrain(),  
BfM\_FreeTrain()

# EduBtM\_DropIndex()

- 파일: EduBtM\_DropIndex.c
- 설명
  - 색인 file에서 B+ tree 색인을 삭제함
    - B+ tree 색인의 root page 및 모든 자식 page들을 각각 deallocate 함

- 파라미터
  - PhysicalFileID \*pFid  
(IN) B+ tree 색인을 삭제할 색인 file의 file ID (= B+ tree 색인 file의 첫 번째 page의 page ID)
  - PageID \*rootPid  
(IN) 삭제할 B+ tree 색인의 root page의 page 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  
edubtm\_FreePages()

# EduBtM\_InsertObject()

- 파일: EduBtM\_InsertObject.c
- 설명
  - B+ tree 색인에 새로운 object를 삽입함
    - edubtm\_Insert()를 호출하여 새로운 object에 대한 <object의 key, object ID> pair를 B+ tree 색인에 삽입함
    - Root page에서 split이 발생하여 새로운 root page 생성이 필요한 경우, edubtm\_root\_insert()를 호출하여 이를 처리함

- 파라미터

- ObjectID \*catObjForFile  
(IN) B+ tree 색인 file 및 색인된 데이터 file에 대한 정보 (sm\_CatOverlayForSysTables) 가 저장된 object의 OID
- PageID \*root  
(IN) B+ tree 색인의 root page의 page ID
- KeyDesc \*kdesc  
(IN) key의 각 속성 값을 서로 구분하기 위한 정보
- KeyValue \*kval  
(IN) 삽입할 object의 key 값
- ObjectID \*oid  
(IN) 삽입할 object의 OID
- Pool \*dlPool  
(INOUT) 새로운 dealloc list element를 할당 받기 위한 pool로서, EduBtM에서는 사용하지 않음
- DeallocListElem \*dlHead  
(INOUT) dealloc list의 첫 번째 element를 가리키고 있는 header로서, EduBtM에서는 사용하지 않음

- 반환값

- Four 에러코드

- 관련 함수

edubtm\_Insert(), edubtm\_root\_insert(), BfM\_GetTrain(), BfM\_FreeTrain()

# EduBtM\_DeleteObject()

- 파일: EduBtM\_DeleteObject.c
- 설명
  - B+ tree 색인에서 object를 삭제함
    - edubtm\_Delete()를 호출하여 삭제할 object에 대한 <object의 key, object ID> pair를 B+ tree 색인에서 삭제함
    - Root page에서 underflow가 발생한 경우, btm\_root\_delete()를 호출하여 이를 처리함
      - btm\_root\_delete()는 underflow가 발생한 root page가 비어있는지를 확인하여 비어있는 경우 root page를 삭제함
    - Root page에서 split이 발생한 경우, edubtm\_root\_insert()를 호출하여 이를 처리함
      - Redistribute 과정 중에 root page의 index entry가 length가 더 긴 index entry로 교체되었을 수 있으므로, root page에서 split이 발생할 수 있음



- 파라미터

- ObjectID \*catObjForFile  
(IN) B+ tree 색인 file 및 색인된 데이터 file에 대한 정보 (sm\_CatOverlayForSysTables) 가 저장된 object의 OID
- PageID \*root  
(IN) B+ tree 색인의 root page의 page ID
- KeyDesc \*kdesc  
(IN) key의 각 속성 값을 서로 구분하기 위한 정보
- KeyValue \*kval  
(IN) 삭제할 object의 key 값
- ObjectID \*oid  
(IN) 삭제할 object의 OID
- 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

edubtm\_root\_insert(), edubtm\_Delete(), btm\_root\_delete(), BfM\_GetTrain(), BfM\_FreeTrain()

# EduBtM\_Fetch()

- 파일: EduBtM\_Fetch.c
- 설명
  - B+ tree 색인에서 검색 조건을 만족하는 첫 번째 object를 검색하고, 검색된 object를 가리키는 cursor를 반환함
    - 파라미터로 주어진 *startCompOp*가 SM\_BOF일 경우,
      - B+ tree 색인의 첫 번째 object (가장 작은 key 값을 갖는 leaf index entry)를 검색함
    - 파라미터로 주어진 *startCompOp*가 SM\_EOF일 경우,
      - B+ tree 색인의 마지막 object (가장 큰 key 값을 갖는 leaf index entry)를 검색함
    - 이외의 경우,
      - *edubtm\_Fetch()*를 호출하여 B+ tree 색인에서 검색 조건을 만족하는 첫 번째 <object의 key, object ID> pair가 저장된 leaf index entry를 검색함
    - 검색된 leaf index entry를 가리키는 cursor를 반환함

- 파라미터

- PageID \*root  
(IN) B+ tree 색인의 root page의 page ID
- KeyDesc \*kdesc  
(IN) key의 각 속성 값을 서로 구분하기 위한 정보
- KeyValue \*startKval  
(IN) 검색 시작 key 값
- Four startCompOp  
(IN) 검색 시작 key 값에 대한 비교 연산자
- KeyValue \*stopKval  
(IN) 검색 종료 key 값
- Four stopCompOp  
(IN) 검색 종료 key 값에 대한 비교 연산자
- BtreeCursor \*cursor  
(OUT) 검색 조건을 만족하는 첫 번째 object에 대응하는 leaf index entry를 가리키는 cursor

- 반환값

- Four 에러코드

- 관련 함수

edubtm\_Fetch(), edubtm\_FirstObject(), edubtm\_LastObject()

# EduBtM\_FetchNext()

- 파일: EduBtM\_FetchNext.c
- 설명
  - B+ tree 색인에서 검색 조건을 만족하는 현재 object의 다음 object를 검색하고, 검색된 object를 가리키는 cursor를 반환함
    - edubtm\_FetchNext()를 호출하여 B+ tree 색인에서 검색 조건을 만족하는 현재 leaf index entry의 다음 leaf index entry를 검색함
    - 검색된 leaf index entry를 가리키는 cursor를 반환함

- 파라미터

- PageID \*root  
(IN) B+ tree 색인의 root page의 page ID
- KeyDesc \*kdesc  
(IN) key의 각 속성 값을 서로 구분하기 위한 정보
- KeyValue \*kval  
(IN) 검색 종료 key 값
- Four compOp  
(IN) 검색 종료 key 값에 대한 비교 연산자
- BtreeCursor \*current  
(IN) 현재 object에 대응하는 leaf index entry를 가리키는 cursor
- BtreeCursor \*next  
(OUT) 검색 조건을 만족하는 다음 object에 대응하는 leaf index entry를 가리키는 cursor

- 반환값

- Four 에러코드

- 관련 함수

edubtm\_FetchNext(), edubtm\_KeyCompare(), BfM\_GetTrain(), BfM\_FreeTrain()

# 구현할 Internal Function 들

- edubtm\_InitLeaf()
- edubtm\_InitInternal()
- edubtm\_FreePages()
- edubtm\_Insert()
- edubtm\_InsertLeaf()
- edubtm\_InsertInternal()
- edubtm\_SplitLeaf()
- edubtm\_SplitInternal()
- edubtm\_root\_insert()
- edubtm\_Delete()
- edubtm\_DeleteLeaf()
- edubtm\_CompactLeafPage()
- edubtm\_CompactInternalPage()  
)
- edubtm\_Fetch()
- edubtm\_FetchNext()
- edubtm\_FirstObject()
- edubtm\_LastObject()
- edubtm\_BinarySearchLeaf()
- edubtm\_BinarySearchInternal()
- edubtm\_KeyCompare()

# edubtm\_InitLeaf()

- 파일: edubtm\_InitPage.c
- 설명
  - Page를 B+ tree 색인의 leaf page로 초기화함
    - Page header를 leaf page로 초기화함
      - pid := 파라미터로 주어진 page ID
      - flags
        - » Page가 B+ tree 색인 page임을 나타내는 bit를 set 함
      - Type
        - » Page가 leaf page임을 나타내는 bit를 set 함
        - » 파라미터로 주어진 *root*가 TRUE인 경우, page가 root page임을 나타내는 bit를 set 함
      - nSlots := 0
      - free := 0
      - prevPage := NIL
      - nextPage := NIL
      - unused := 0

- 파라미터
  - PageID \*leaf  
(IN) 초기화할 page의 page ID
  - Boolean root  
(IN) 초기화할 page가 root page임을 나타내는 flag
  - Boolean isTmp  
(IN) 초기화할 page가 temporary page임을 나타내는 flag로서, EduBtM에  
서는 사용하지 않음
- 반환값
  - Four 에러코드
- 관련 함수  
BfM\_GetNewTrain(), BfM\_FreeTrain(), BfM\_SetDirty()



# edubtm\_InitInternal()

- 파일: edubtm\_InitPage.c
- 설명
  - Page를 B+ tree 색인의 internal page로 초기화함
    - Page header를 internal page로 초기화함
      - *pid* := 파라미터로 주어진 page ID
      - *flags*
        - » Page가 B+ tree 색인 page임을 나타내는 bit를 set 함
      - *type*
        - » Page가 internal page임을 나타내는 bit를 set 함
        - » 파라미터로 주어진 *root*가 TRUE인 경우, page가 root page임을 나타내는 bit를 set 함
      - *p0* := NIL
      - *nSlots* := 0
      - *free* := 0
      - *unused* := 0

- 파라미터
  - PageID   \*internal  
  (IN) 초기화할 page의 page ID
  - Boolean   root  
  (IN) 초기화할 page가 root page임을 나타내는 flag
  - Boolean   isTmp  
  (IN) 초기화할 page가 temporary page임을 나타내는 flag로서, EduBtM에  
  서는 사용하지 않음
  
- 반환값
  - Four   에러코드
  
- 관련 함수
  - BfM\_GetNewTrain(), BfM\_FreeTrain(), BfM\_SetDirty()

# edubtm\_FreePages()

- 파일: edubtm\_FreePages.c
- 설명
  - B+ tree 색인 page를 deallocate 함
    - 파라미터로 주어진 page의 모든 자식 page들에 대해 재귀적으로 edubtm\_FreePages()를 호출하여 해당 page들을 deallocate 함
    - 파라미터로 주어진 page를 deallocate 함
      - Page header의 *type*에서 해당 page가 deallocate 될 page임을 나타내는 bit를 set 및 나머지 bit들을 unset 함
      - 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

- 파라미터

- PhysicalFileID \*pFid  
(IN) deallocate 할 page가 속한 색인 file의 file ID (= 색인 file의 첫 번째 page의 page ID)
- PageID \*curPid  
(IN) deallocate 할 page의 page 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

BfM\_GetNewTrain(), BfM\_FreeTrain(), BfM\_SetDirty(),  
Util\_getElementFromPool()

# edubtm\_Insert()

- 파일: edubtm\_Insert.c
- 설명
  - 파라미터로 주어진 page를 root page로 하는 B+ tree 색인에 새로운 object에 대한 <object의 key, object ID> pair를 삽입하고, root page에서 split이 발생한 경우, split으로 생성된 새로운 page를 가리키는 internal index entry를 반환함
    - 파라미터로 주어진 root page가 internal page인 경우,
      - 새로운 <object의 key, object ID> pair를 삽입할 leaf page를 찾기 위해 다음으로 방문할 자식 page를 결정함
      - 결정된 자식 page를 root page로 하는 B+ subtree에 새로운 <object의 key, object ID> pair를 삽입하기 위해 재귀적으로 edubtm\_Insert()를 호출함

- 결정된 자식 page에서 split이 발생한 경우, 해당 split으로 생성된 새로운 page를 가리키는 internal index entry를 파라미터로 주어진 root page에 삽입함
  - » 해당 index entry의 삽입 위치 (slot 번호) 를 결정함
    - Slot array에 저장된 index entry의 offset들이 index entry의 key 순으로 정렬되어야 함
  - » edubtm\_InsertInternal()을 호출하여 결정된 slot 번호로 index entry를 삽입함
- 파라미터로 주어진 root page에서 split이 발생한 경우, 해당 split으로 생성된 새로운 page를 가리키는 internal index entry를 반환함
- 파라미터로 주어진 root page가 leaf page인 경우,
  - edubtm\_InsertLeaf()를 호출하여 해당 page에 새로운 <object의 key, object ID> pair를 삽입함
  - Split이 발생한 경우, 해당 split으로 생성된 새로운 page를 가리키는 internal index entry를 반환함

- 파라미터

- ObjectID \*catObjForFile  
(IN) B+ tree 색인 file 및 색인된 데이터 file에 대한 정보 (sm\_CatOverlayForSysTables) 가 저장된 object의 OID
- PageID \*root  
(IN) B+ tree 색인의 root page의 page ID
- KeyDesc \*kdesc  
(IN) key의 각 속성 값을 서로 구분하기 위한 정보
- KeyValue \*kval  
(IN) 삽입할 object의 key 값
- ObjectID \*oid  
(IN) 삽입할 object의 OID
- Boolean \*f  
(OUT) root page가 merge 되었음을 나타내는 flag로서, EduBtM에서는 사용하지 않음
- Boolean \*h  
(OUT) root page가 split 되었음을 나타내는 flag
- InternalItem \*item  
(OUT) root page split으로 생성된 새로운 page를 가리키는 internal index entry
- Pool \*dlPool  
(INOUT) 새로운 dealloc list element를 할당 받기 위한 pool로서, EduBtM에서는 사용하지 않음
- DeallocListElem \*dlHead  
(INOUT) dealloc list의 첫 번째 element를 가리키고 있는 header로서, EduBtM에서는 사용하지 않음

- 반환값

- Four 에러코드

- 관련 함수

edubtm\_InsertLeaf(), edubtm\_InsertInternal(), edubtm\_BinarySearchInternal(), BfM\_GetTrain(), BfM\_FreeTrain(), BfM\_SetDirty()

# edubtm\_InsertLeaf()

- 파일: edubtm\_Insert.c

- 설명

- Leaf page에 새로운 index entry를 삽입하고, split이 발생한 경우, split으로 생성된 새로운 leaf page를 가리키는 internal index entry를 반환함
  - 새로운 index entry의 삽입 위치 (slot 번호) 를 결정함
    - Slot array에 저장된 index entry의 offset들이 index entry의 key 순으로 정렬되어야 함
    - 새로운 index entry의 key 값과 동일한 key 값을 갖는 index entry가 존재하는 경우 eDUPLICATEDKEY\_BTМ error 를 반환함
  - 새로운 index entry 삽입을 위해 필요한 자유 영역의 크기를 계산함
    - Align 된 key 영역을 고려한 새로운 index entry의 크기 + slot의 크기
  - Page에 여유 영역이 있는 경우,
    - 필요시 page를 compact 함
    - 결정된 slot 번호로 새로운 index entry를 삽입함
      - » Page의 contiguous free area에 새로운 index entry를 복사함
      - » 결정된 slot 번호를 갖는 slot을 사용하기 위해 slot array를 재배열함
      - » 결정된 slot 번호를 갖는 slot에 새로운 index entry의 offset을 저장함
      - » Page의 header를 갱신함
  - Page에 여유 영역이 없는 경우 (page overflow),
    - edubtm\_SplitLeaf()를 호출하여 page를 split 함
    - Split으로 생성된 새로운 leaf page를 가리키는 internal index entry를 반환함



- 파라미터
  - ObjectID \*catObjForFile  
(IN) B+ tree 색인 file 및 색인된 데이터 file에 대한 정보 (sm\_CatOverlayForSysTables) 가 저장된 object의 OID
  - PageID \*pid  
(IN) index entry를 삽입할 leaf page의 page ID
  - BtreeLeaf \*page  
(INOUT) index entry를 삽입할 leaf page가 저장된 buffer element에 대한 포인터
  - KeyDesc \*kdesc  
(IN) key의 각 속성 값을 서로 구분하기 위한 정보
  - KeyValue \*kval  
(IN) 삽입할 index entry의 key 값
  - ObjectID \*oid  
(IN) 삽입할 index entry에 저장된 object ID (OID)
  - Boolean \*f  
(OUT) page가 merge 되었음을 나타내는 flag로서, EduBtM에서는 사용하지 않음
  - Boolean \*h  
(OUT) page가 split 되었음을 나타내는 flag
  - InternalItem \*item  
(OUT) page split으로 생성된 새로운 leaf page를 가리키는 internal index entry
- 반환값
  - Four 에러코드
- 관련 함수
  - edubtm\_SplitLeaf(), edubtm\_CompactLeafPage(), edubtm\_BinarySearchLeaf()

# edubtm\_InsertInternal()

- 파일: edubtm\_Insert.c
- 설명
  - Internal page에 새로운 index entry를 삽입하고, split이 발생한 경우, split으로 생성된 새로운 internal page를 가리키는 internal index entry를 반환함
    - 새로운 index entry 삽입을 위해 필요한 자유 영역의 크기를 계산함
      - Align 된 key 영역을 고려한 새로운 index entry의 크기 + slot의 크기
    - Page에 여유 영역이 있는 경우,
      - 필요시 page를 compact 함
      - 파라미터로 주어진 slot 번호의 다음 slot 번호로 새로운 index entry를 삽입함
        - » Page의 contiguous free area에 새로운 index entry를 복사함
        - » 결정된 slot 번호를 갖는 slot을 사용하기 위해 slot array를 재배열함
        - » 결정된 slot 번호를 갖는 slot에 새로운 index entry의 offset을 저장함
        - » Page의 header를 갱신함
    - Page에 여유 영역이 없는 경우 (page overflow),
      - edubtm\_SplitInternal()을 호출하여 page를 split 함
      - Split으로 생성된 새로운 internal page를 가리키는 internal index entry를 반환함

- 파라미터

- ObjectID \*catObjForFile  
(IN) B+ tree 색인 file 및 색인된 데이터 file에 대한 정보 (sm\_CatOverlayForSysTables) 가 저장된 object의 OID
- BtreeInternal \*page  
(INOUT) index entry를 삽입할 internal page가 저장된 buffer element에 대한 포인터
- InternalItem \*item  
(IN) 삽입할 index entry (align 되지 않음)
- Two high  
(IN) 삽입할 index entry의 key 값보다 작은 key 값을 갖는 index entry들 중 가장 큰 key 값을 갖는 index entry의 slot 번호
- Boolean \*h  
(OUT) page가 split 되었음을 나타내는 flag
- InternalItem \*ritem  
(OUT) page split으로 생성된 새로운 internal page를 가리키는 internal index entry

- 반환값

- Four 에러코드

- 관련 함수

edubtm\_SplitInternal(), edubtm\_CompactInternalPage()

# edubtm\_SplitLeaf()

- 파일: edubtm\_Split.c
- 설명
  - Overflow가 발생한 leaf page를 split 하여 파라미터로 주어진 index entry를 삽입하고, split으로 생성된 새로운 leaf page를 가리키는 internal index entry를 반환함
    - 새로운 page를 할당 받음
    - 할당 받은 page를 leaf page로 초기화함

- 기존 index entry들 및 삽입할 index entry를 key 순으로 정렬하여 overflow가 발생한 page 및 할당 받은 page에 나누어 저장함
  - 먼저, overflow가 발생한 page에 데이터 영역을 50% 이상 채우는 수의 index entry들을 저장함
  - 나머지 index entry들을 할당 받은 page에 저장함
  - 각 page의 header를 갱신함
- 할당 받은 page를 leaf page들간의 doubly linked list에 추가함
  - 할당 받은 page가 overflow가 발생한 page의 다음 page가 되도록 추가함
- 할당 받은 page를 가리키는 internal index entry를 생성함
  - Discriminator key 값 := 할당 받은 page의 첫 번째 index entry (slot 번호 = 0) 의 key 값
    - » B+ tree 색인에서는 internal index entry의 key 값 (discriminator key 값) 이 leaf index entry의 key 값과 중복될 수 있음
  - 자식 page의 번호 := 할당 받은 page의 번호
- Split된 page가 ROOT일 경우, type을 LEAF로 변경함
- 생성된 index entry를 반환함

- 파라미터

- ObjectID \*catObjForFile  
(IN) B+ tree 색인 file 및 색인된 데이터 file에 대한 정보 (sm\_CatOverlayForSysTables) 가 저장된 object의 OID
- PageID \*root  
(IN) overflow가 발생한 page의 page ID
- BtreeLeaf \*fpage  
(INOUT) overflow가 발생한 page가 저장된 buffer element에 대한 포인터
- Two high  
(IN) 삽입할 index entry의 key 값보다 작은 key 값을 갖는 index entry들 중 가장 큰 key 값을 갖는 index entry의 slot 번호
- LeafItem \*item  
(IN) 삽입할 index entry (align 되지 않음)
- InternalItem \*ritem  
(OUT) page split으로 생성된 새로운 leaf page를 가리키는 internal index entry

- 반환값

- Four 에러코드

- 관련 함수

edubtm\_InitLeaf(), edubtm\_CompactLeafPage(), btm\_AllocPage(), BfM\_GetTrain(), BfM\_GetNewTrain(), BfM\_FreeTrain(), BfM\_SetDirty()

# edubtm\_SplitInternal()

- 파일: edubtm\_Split.c
- 설명
  - Overflow가 발생한 internal page를 split 하여 파라미터로 주어진 index entry를 삽입하고, split으로 생성된 새로운 internal page를 가리키는 internal index entry를 반환함
    - 새로운 page를 할당 받음
    - 할당 받은 page를 internal page로 초기화함
    - 기존 index entry들 및 삽입할 index entry를 key 순으로 정렬하여 overflow가 발생한 page 및 할당 받은 page에 나누어 저장함
      - 먼저, overflow가 발생한 page에 데이터 영역을 50% 이상 채우는 수의 index entry들을 저장함
      - 할당 받은 page의 header의 p0 변수에 아직 저장되지 않은 index entry 중 첫 번째 index entry (1<sup>st</sup> entry) 가 가리키는 자식 page의 번호를 저장함
      - 1<sup>st</sup> entry는 할당 받은 page를 가리키는 internal index entry로 설정하여 반환함
        - » 자식 page의 번호 := 할당 받은 page의 번호
      - 나머지 index entry들을 할당 받은 page에 저장함
      - 각 page의 header를 갱신함
    - Split된 page가 ROOT일 경우, type을 INTERNAL로 변경함

- 파라미터

- ObjectID \*catObjForFile  
(IN) B+ tree 색인 file 및 색인된 데이터 file에 대한 정보 (sm\_CatOverlayForSysTables) 가 저장된 object의 OID
- BtreeLeaf \*fpage  
(INOUT) overflow가 발생한 page가 저장된 buffer element에 대한 포인터
- Two high  
(IN) 삽입할 index entry의 key 값보다 작은 key 값을 갖는 index entry들 중 가장 큰 key 값을 갖는 index entry의 slot 번호
- LeafItem \*item  
(IN) 삽입할 index entry (align 되지 않음)
- InternalItem \*ritem  
(OUT) page split으로 생성된 새로운 internal page를 가리키는 internal index entry

- 반환값

- Four 에러코드

- 관련 함수

edubtm\_InitInternal(), edubtm\_CompactInternalPage(), btm\_AllocPage(),  
BfM\_GetNewTrain(), BfM\_FreeTrain(), BfM\_SetDirty()



# edubtm\_root\_insert()

- 파일: edubtm\_root.c
- 설명
  - Root page가 split 된 B+ tree 색인을 위한 새로운 root page를 생성함
    - 새로운 page를 할당 받음
    - 기존 root page를 할당 받은 page로 복사함
    - 기존 root page를 새로운 root page로서 초기화함
      - B+ tree 색인의 root page의 page ID를 일관되게 유지하기 위함
    - 할당 받은 page와 root page split으로 생성된 page가 새로운 root page의 자식 page들이 되도록 설정함
      - Split으로 생성된 page를 가리키는 internal index entry를 새로운 root page에 삽입함
      - 새로운 root page의 header의 p0 변수에 할당 받은 page의 번호를 저장함
      - 새로운 root page의 두 자식 page들이 leaf인 경우, 두 자식 page들간의 doubly linked list를 설정함
        - » Split으로 생성된 page가 할당 받은 page의 다음 page가 되도록 설정함

- 파라미터

- ObjectID \*catObjForFile  
(IN) B+ tree 색인 file 및 색인된 데이터 file에 대한 정보 (sm\_CatOverlayForSysTables) 가 저장된 object의 OID
- PageID \*root  
(IN) split 된 root page의 page ID
- InternalItem \*ritem  
(IN) root page split으로 생성된 새로운 page를 가리키는 internal index entry

- 반환값

- Four 에러코드

- 관련 함수

edubtm\_InitInternal(), btm\_AllocPage(), BfM\_GetTrain(),  
BfM\_GetNewTrain(), BfM\_FreeTrain(), BfM\_SetDirty()

# edubtm\_Delete()

- 파일: edubtm\_Delete.c
- 설명
  - 파라미터로 주어진 page를 root page로 하는 B+ tree 색인에서 <object의 key, object ID> pair를 삭제함
    - 파라미터로 주어진 root page가 internal page인 경우,
      - 삭제할 <object의 key, object ID> pair가 저장된 leaf page를 찾기 위해 다음으로 방문할 자식 page를 결정함
      - 결정된 자식 page를 root page로 하는 B+ subtree에서 <object의 key, object ID> pair를 삭제하기 위해 재귀적으로 edubtm\_Delete()를 호출함
      - 결정된 자식 page에서 underflow가 발생한 경우, btm\_Underflow()를 호출하여 이를 처리함
        - » Underflow가 발생한 자식 page의 부모 page (파라미터로 주어진 root page) 에서 overflow가 발생한 경우, edubtm\_InsertInternal()을 호출하여 overflow로 인해 삽입되지 못한 internal index entry를 부모 page에 삽입함
          - edubtm\_InsertInternal() 호출 결과로서 부모 page가 split 되므로, out parameter인 *h*를 TRUE로 설정하고 split으로 생성된 새로운 page를 가리키는 internal index entry를 반환함
        - » btm\_Underflow() 호출 결과로서 파라미터로 주어진 root page의 내용이 변경되므로, btm\_Underflow() 호출 후 root page의 DIRTY bit를 1로 set 해야 함
    - 파라미터로 주어진 root page가 leaf page인 경우,
      - edubtm\_DeleteLeaf()를 호출하여 해당 page에서 <object의 key, object ID> pair를 삭제함
      - 해당 page에서 underflow가 발생한 경우 (page의 data 영역 중 자유 영역의 크기 > (page의 data 영역의 전체 크기 / 2)), out parameter인 *h*를 TRUE로 설정함

- 파라미터
  - ObjectID \*catObjForFile  
(IN) B+ tree 색인 file 및 색인된 데이터 file에 대한 정보 (sm\_CatOverlayForSysTables) 가 저장된 object의 OID
  - PageID \*root  
(IN) B+ tree 색인의 root page의 page ID
  - KeyDesc \*kdesc  
(IN) key의 각 속성 값을 서로 구분하기 위한 정보
  - KeyValue \*kval  
(IN) 삭제할 object의 key 값
  - ObjectID \*oid  
(IN) 삭제할 object의 OID
  - Boolean \*f  
(OUT) root page에서 underflow가 발생했음을 나타내는 flag
  - Boolean \*h  
(OUT) root page가 split 되었음을 나타내는 flag
  - InternalItem \*item  
(OUT) root page split으로 생성된 새로운 page를 가리키는 internal index entry
  - 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  
edubtm\_InsertInternal(), edubtm\_DeleteLeaf(), edubtm\_BinarySearchInternal(), btm\_Underflow(), BfM\_GetTrain(), BfM\_FreeTrain(), BfM\_SetDirty()

# edubtm\_DeleteLeaf()

- 파일: edubtm\_Delete.c
- 설명
  - Leaf page에서 <object의 key, object ID> pair를 삭제함
    - 삭제할 <object의 key, object ID> pair가 저장된 index entry의 offset이 저장된 slot을 삭제함
      - Slot array 중간에 삭제된 빈 slot이 없도록 slot array를 compact 함
    - Leaf page의 header를 갱신함
    - Leaf page에서 underflow가 발생한 경우(page의 data 영역 중 자유 영역의 크기 > (page의 data 영역의 전체 크기 / 2)), out parameter인 *isUnderflow*를 TRUE로 설정함

- 파라미터

- PhysicalFileID \*pFid  
(IN) B+ tree 색인 file의 file ID (= B+ tree 색인 file의 첫 번째 page의 page ID)
- PageID \*pid  
(IN) index entry를 삭제할 leaf page의 page ID
- BtreeLeaf \*apage  
(INOUT) index entry를 삭제할 leaf page가 저장된 buffer element에 대한 포인터
- KeyDesc \*kdesc  
(IN) key의 각 속성 값을 서로 구분하기 위한 정보
- KeyValue \*kval  
(IN) 삭제할 index entry의 key 값
- ObjectID \*oid  
(IN) 삭제할 index entry에 저장된 object ID (OID)
- Boolean \*f  
(OUT) leaf page에서 underflow가 발생했음을 나타내는 flag
- Boolean \*h  
(OUT) leaf page가 split 되었음을 나타내는 flag로서, EduBtM에서는 사용하지 않음
- InternalItem \*item  
(OUT) leaf page split으로 생성된 새로운 page를 가리키는 internal index entry로서, EduBtM에서는 사용하지 않음
- Pool \*dlPool  
(INOUT) 새로운 dealloc list element를 할당 받기 위한 pool로서, EduBtM에서는 사용하지 않음
- DeallocListElem \*dlHead  
(INOUT) dealloc list의 첫 번째 element를 가리키고 있는 header로서, EduBtM에서는 사용하지 않음

- 반환값

- Four 에러코드

- 관련 함수

edubtm\_BinarySearchLeaf(), btm\_ObjectIdComp(), BfM\_SetDirty()

# edubtm\_CompactLeafPage()

- 파일: edubtm\_Compact.c
- 설명
  - Leaf page의 데이터 영역의 모든 자유 영역이 연속된 하나의 contiguous free area를 형성하도록 index entry들의 offset를 조정함
    - 파라미터로 주어진 *slotNo*가 NIL이 아닌 경우,
      - *slotNo*에 대응하는 index entry를 제외한 page의 모든 index entry들을 데이터 영역의 가장 앞부분부터 연속되게 저장함
        - » Index entry 저장 순서: 대응하는 slot 번호 순
      - *slotNo*에 대응하는 index entry를 데이터 영역 상에서의 마지막 index entry로 저장함
    - 파라미터로 주어진 *slotNo*가 NIL인 경우,
      - Page의 모든 index entry들을 데이터 영역의 가장 앞부분부터 연속되게 저장함
        - » Index entry 저장 순서: 대응하는 slot 번호 순
    - Page header를 갱신함

- 파라미터
  - BtreeLeaf \*apage  
(INOUT) compact할 leaf page가 저장된 buffer element에 대한 포인터
  - Two slotNo  
(IN) page의 데이터 영역 상에서 마지막으로 저장할 index entry의 slot 번호
- 반환값
  - void
- 관련 함수 없음



# edubtm\_CompactInternalPage()

- 파일: edubtm\_Compact.c
- 설명
  - Internal page의 데이터 영역의 모든 자유 영역이 연속된 하나의 contiguous free area를 형성하도록 index entry들의 offset를 조정함
    - 파라미터로 주어진 *slotNo*가 NIL이 아닌 경우,
      - *slotNo*에 대응하는 index entry를 제외한 page의 모든 index entry들을 데이터 영역의 가장 앞부분부터 연속되게 저장함
        - » Index entry 저장 순서: 대응하는 slot 번호 순
      - *slotNo*에 대응하는 index entry를 데이터 영역 상에서의 마지막 index entry로 저장함
    - 파라미터로 주어진 *slotNo*가 NIL인 경우,
      - Page의 모든 index entry들을 데이터 영역의 가장 앞부분부터 연속되게 저장함
        - » Index entry 저장 순서: 대응하는 slot 번호 순
    - Page header를 갱신함

- 파라미터
  - BtreeInternal \*apage  
(INOUT) compact할 internal page가 저장된 buffer element에 대한 포인터
  - Two slotNo  
(IN) page의 데이터 영역 상에서 마지막으로 저장할 index entry의 slot 번호
- 반환값
  - void
- 관련 함수 없음

# edubtm\_Fetch()

- 파일: EduBtM\_Fetch.c

- 설명

- 파라미터로 주어진 page를 root page로 하는 B+ tree 색인에서 검색 조건을 만족하는 첫 번째 <object의 key, object ID> pair가 저장된 leaf index entry를 검색하고, 검색된 leaf index entry를 가리키는 cursor를 반환함. 첫 번째 object는, 검색 조건을 만족하는 object들 중 검색 시작 key 값과 가장 가까운 key 값을 가지는 object를 의미함
  - 파라미터로 주어진 root page가 internal page인 경우,
    - 검색 조건을 만족하는 첫 번째 <object의 key, object ID> pair가 저장된 leaf page를 찾기 위해 다음으로 방문할 자식 page를 결정함
    - 결정된 자식 page를 root page로 하는 B+ subtree에서 검색 조건을 만족하는 첫 번째 <object의 key, object ID> pair가 저장된 leaf index entry를 검색하기 위해 재귀적으로 edubtm\_Fetch()를 호출함
    - 검색된 leaf index entry를 가리키는 cursor를 반환함

- 파라미터로 주어진 root page가 leaf page인 경우,
  - 검색 조건을 만족하는 첫 번째 <object의 key, object ID> pair가 저장된 index entry를 검색함
    - » 검색 조건은 파라미터로 주어진 검색 시작/종료 key 값 및 비교 연산자를 통해 알 수 있음
      - SM\_EQ: 검색 시작/종료 key 값과 같은
      - SM\_LT: 검색 시작/종료 key 값보다 작은
      - SM\_LE: 검색 시작/종료 key 값보다 작거나 같은
      - SM\_GT: 검색 시작/종료 key 값보다 큰
      - SM\_GE: 검색 시작/종료 key 값보다 크거나 같은
    - » 검색 종료 연산자가 SM\_LT, SM\_LE, SM\_GT, SM\_GE, SM\_EQ 중 하나일 때에만, 검색 시작 조건을 이용해 찾은 key 값과 검색 종료 조건을 비교하여 검색함
  - 검색된 index entry를 가리키는 cursor를 반환함

- 파라미터

- PageID \*root  
(IN) B+ tree 색인의 root page의 page ID
- KeyDesc \*kdesc  
(IN) key의 각 속성 값을 서로 구분하기 위한 정보
- KeyValue \*startKval  
(IN) 검색 시작 key 값
- Four startCompOp  
(IN) 검색 시작 key 값에 대한 비교 연산자
- KeyValue \*stopKval  
(IN) 검색 종료 key 값
- Four stopCompOp  
(IN) 검색 종료 key 값에 대한 비교 연산자
- BtreeCursor \*cursor  
(OUT) 검색 조건을 만족하는 첫 번째 <object의 key, object ID> pair가 저장된 leaf index entry를 가리키는 cursor

- 반환값

- Four 에러코드

- 관련 함수

edubtm\_BinarySearchLeaf(), edubtm\_BinarySearchInternal(), edubtm\_KeyCompare(), BfM\_GetTrain(), BfM\_FreeTrain()

# edubtm\_FetchNext()

- 파일: EduBtM\_FetchNext.c

- 설명

- B+ tree 색인에서 검색 조건을 만족하는 현재 leaf index entry의 다음 leaf index entry를 검색하고, 검색된 leaf index entry를 가리키는 cursor를 반환함. 검색 조건이 SM\_GT, SM\_GE, SM\_BOF 일 경우 key 값이 작아지는 방향으로 backward scan을 하며, 그 외의 경우 key 값이 커지는 방향으로 forward scan을 한다
  - 검색 조건을 만족하는 다음 leaf index entry를 검색함
    - 검색 조건은 파라미터로 주어진 검색 종료 key 값 및 비교 연산자를 통해 알 수 있음 (아래의 검색 조건들 이외는 고려하지 않음)
      - » SM\_LT: 검색 종료 key 값보다 작은
      - » SM\_LE: 검색 종료 key 값보다 작거나 같은
      - » SM\_GT: 검색 종료 key 값보다 큰
      - » SM\_GE: 검색 종료 key 값보다 크거나 같은
  - 검색된 leaf index entry를 가리키는 cursor를 반환함

- 파라미터

- KeyDesc \*kdesc  
(IN) key의 각 속성 값을 서로 구분하기 위한 정보
- KeyValue \*kval  
(IN) 검색 종료 key 값
- Four compOp  
(IN) 검색 종료 key 값에 대한 비교 연산자
- BtreeCursor \*current  
(IN) 검색 조건을 만족하는 현재 leaf index entry를 가리키는 cursor
- BtreeCursor \*next  
(OUT) 검색 조건을 만족하는 다음 leaf index entry를 가리키는 cursor

- 반환값

- Four 에러코드

- 관련 함수

edubtm\_KeyCompare(), BfM\_GetTrain(), BfM\_FreeTrain()

# edubtm\_FirstObject()

- 파일: edubtm\_FirstObject.c
- 설명
  - B+ tree 색인에서 첫 번째 object (가장 작은 key 값을 갖는 leaf index entry) 를 검색함
    - B+ tree 색인의 첫 번째 leaf page의 첫 번째 leaf index entry를 가리키는 cursor를 반환함
    - 검색 종료 key 값이 첫 번째 object의 key 값 보다 작거나, key 값은 같으나 검색 종료 연산이 SM\_LT 인 경우 CURSOR\_EOS 반환



- 파라미터

- PageID \*root  
(IN) B+ tree 색인의 root page의 page ID
- KeyDesc \*kdesc  
(IN) key의 각 속성 값을 서로 구분하기 위한 정보
- KeyValue \*stopKval  
(IN) 검색 종료 key 값
- Four stopCompOp  
(IN) 검색 종료 key 값에 대한 비교 연산자
- BtreeCursor \*cursor  
(OUT) B+ tree 색인의 첫 번째 object에 대응하는 leaf index entry를 가리키는 cursor

- 반환값

- Four 에러코드

- 관련 함수

edubtm\_KeyCompare(), BfM\_GetTrain(), BfM\_FreeTrain()

# edubtm\_LastObject()

- 파일: edubtm\_LastObject.c
- 설명
  - B+ tree 색인에서 마지막 object (가장 큰 key값을 갖는 leaf index entry) 를 검색함
    - B+ tree 색인의 마지막 leaf page의 마지막 index entry (slot 번호 =  $nSlots - 1$ ) 를 가리키는 cursor를 반환함
    - 검색 종료 key 값이 마지막 object의 key 값 보다 크거나, key 값은 같으나 검색 종료 연산이 SM\_GT인 경우 CURSOR\_EOS 반환

- 파라미터

- PageID \*root  
(IN) B+ tree 색인의 root page의 page ID
- KeyDesc \*kdesc  
(IN) key의 각 속성 값을 서로 구분하기 위한 정보
- KeyValue \*stopKval  
(IN) 검색 종료 key 값
- Four stopCompOp  
(IN) 검색 종료 key 값에 대한 비교 연산자
- BtreeCursor \*cursor  
(OUT) B+ tree 색인의 마지막 object에 대응하는 leaf index entry를 가리키는 cursor

- 반환값

- Four 에러코드

- 관련 함수

edubtm\_KeyCompare(), BfM\_GetTrain(), BfM\_FreeTrain()

# edubtm\_BinarySearchLeaf()

- 파일: edubtm\_BinarySearch.c
- 설명
  - Leaf page에서 파라미터로 주어진 key 값보다 작거나 같은 key 값을 갖는 index entry를 검색하고, 검색된 index entry의 위치 (slot 번호) 를 반환함
    - 파라미터로 주어진 key 값과 같은 key 값을 갖는 index entry가 존재하는 경우,
      - 해당 index entry의 slot 번호 및 TRUE를 반환함
    - 파라미터로 주어진 key 값과 같은 key 값을 갖는 index entry가 존재하지 않는 경우,
      - 파라미터로 주어진 key 값보다 작은 key 값을 갖는 index entry들 중 가장 큰 key 값을 갖는 index entry의 slot 번호 및 FALSE를 반환함. 주어진 key 값보다 작은 key 값을 갖는 entry가 없을 경우 slot 번호로 -1을 반환함.

- 파라미터
  - BtreeLeaf \*lpage  
(IN) B+ tree 색인의 root page의 page ID
  - KeyDesc \*kdesc  
(IN) key의 각 속성 값을 서로 구분하기 위한 정보
  - KeyValue \*kval  
(IN) 검색할 key 값
  - Two \*idx  
(OUT) 검색된 index entry의 slot 번호
- 반환값
  - Four TRUE 또는 FALSE
- 관련 함수  
edubtm\_KeyCompare()

# edubtm\_BinarySearchInternal()

- 파일: edubtm\_BinarySearch.c
- 설명
  - Internal page에서 파라미터로 주어진 key 값보다 작거나 같은 key 값을 갖는 index entry를 검색하고, 검색된 index entry의 위치 (slot 번호) 를 반환함
    - 파라미터로 주어진 key 값과 같은 key 값을 갖는 index entry가 존재하는 경우,
      - 해당 index entry의 slot 번호 및 TRUE를 반환함
    - 파라미터로 주어진 key 값과 같은 key 값을 갖는 index entry가 존재하지 않는 경우,
      - 파라미터로 주어진 key 값보다 작은 key 값을 갖는 index entry들 중 가장 큰 key 값을 갖는 index entry의 slot 번호 및 FALSE를 반환함. 주어진 key 값보다 작은 key 값을 갖는 entry가 없을 경우 slot 번호로 -1을 반환함.

- 파라미터
  - BtreeInternal \*ipage  
(IN) B+ tree 색인의 root page의 page ID
  - KeyDesc \*kdesc  
(IN) key의 각 속성 값을 서로 구분하기 위한 정보
  - KeyValue \*kval  
(IN) 검색할 key 값
  - Two \*idx  
(OUT) 검색된 index entry의 slot 번호
- 반환값
  - Four TRUE 또는 FALSE
- 관련 함수  
edubtm\_KeyCompare()

# edubtm\_KeyCompare()

- 파일: edubtm\_Compare.c
- 설명
  - 파라미터로 주어진 두 key 값의 대소를 비교하고, 비교 결과를 반환함. Variable length string의 경우 사전식 순서를 이용하여 비교함.
  - 두 key 값이 같은 경우, EQUAL을 반환함
  - 첫 번째 key 값이 큰 경우, GREATER를 반환함
  - 첫 번째 key 값이 작은 경우, LESS를 반환함



- 파라미터
  - KeyDesc \*kdesc  
(IN) key의 각 속성 값을 서로 구분하기 위한 정보
  - KeyValue \*key1  
(IN) 비교할 첫 번째 key 값
  - KeyValue \*key2  
(IN) 비교할 두 번째 key 값
- 반환값
  - Four EQUAL, GREATER, 또는 LESS
- 관련 함수 없음

# 제공되는 API Function 들

- BfM\_GetTrain()
  - Page/train을 buffer element에 fix 하고, 해당 buffer element에 대한 포인터를 반환함
  - 파라미터
    - TrainID \*trainId  
(IN) Fix 할 page의 page ID 또는 train의 첫 번째 page의 page ID
    - Char \*\*retBuf  
(OUT) Fix 된 page/train이 저장될 buffer element에 대한 포인터
    - Four type  
(IN) Buffer의 type
  - 반환값
    - Four 에러코드

– 예

```
Four edubtm_Fetch(  
    PageID *root, /* IN: ID of the current root page of the subtree */  
    ...)  
{  
    BtreePage *apage;      /* pointer to a buffer */  
    ...  
    /* Fix the page to the buffer */  
    e = BfM_GetTrain(root, (char **)&apage, PAGE_BUF); if (e < 0)  
    ERR(e);  
    ...  
}
```

- BfM\_GetNewTrain()
  - Disk 상에서 할당되지 않은 새로운 page/train을 buffer element에 fix 하고, 해당 buffer element에 대한 포인터를 반환함
  - 파라미터
    - TrainID \*trainId  
(IN) Fix 할 page의 page ID 또는 train의 첫 번째 page의 page ID
    - Char \*\*retBuf  
(OUT) Fix 된 page/train이 저장될 buffer element에 대한 포인터
    - Four type  
(IN) Buffer의 type
  - 반환값
    - Four 에러코드

– 예

```
Four edubtm_InitInternal(
    PageID *internal,      /* IN: ID of the page to be initialized */
    ...)
{
    BtreeInternal *page;   /* pointer to a buffer */
    ...
    /* Fix the page that has been newly allocated on the
    disk to the buffer */ e = BfM_GetNewTrain(internal,
    (char **)&page, PAGE_BUF);
    if (e < 0) ERR(e);
    ...
}
```

- BfM\_FreeTrain()
  - Page/train을 buffer element에서 unfix 함
  - 파라미터
    - TrainID    \*trainId  
 (IN) Unfix 할 page의 page ID 또는 train의 첫 번째 page의 page ID
    - Four    type  
 (IN) Buffer의 type
  - 반환값
    - Four    에러코드

## — 예

```
Four edubtm_Fetch(  
    PageID *root,      /* IN: ID of the current root page of the subtree */  
    ...)  
{  
    ...  
    /* Unfix the page from the buffer */  
    e = BfM_FreeTrain(root, PAGE_BUF);  
    if (e < 0) ERR(e);  
    ...  
}
```

- BfM\_SetDirty()
  - Buffer element에 저장된 page/train이 수정되었음을 표시하기 위해 DIRTY bit를 set 함
  - 파라미터
    - TrainID    \*trainId  
 (IN) DIRTY bit를 set 할 page의 page ID 또는 train의 첫 번째 page의 page ID
    - Four    type  
 (IN) Buffer의 type
  - 반환값
    - Four    에러코드



## – 예

```
Four edubtm_InitInternal(
    PageID *internal, /* IN: ID of the page to be initialized */
    ...)
{
    ...
    /* Set the DIRTY bit */
    e = BfM_SetDirty(internal,
        PAGE_BUF); if (e < 0) ERRB1(e,
        internal, PAGE_BUF);
    ...
}
```

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

## — 예

```
Four edubtm_FreePages(...
    PageID    *curPid,    /* IN: ID of the page to be freed */
    Pool      *dlPool,    /* INOUT: pool of the elements of the dealloc list */
    DeallocListElem *dlHead) /* INOUT: head of the dealloc list */
{
    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 = *curPid;
    dlElem->next = dlHead->next;
    dlHead->next = dlElem;
    ...
}
```

# 제공되는 Function 들

- btm\_AllocPage()
  - B+ tree 색인 page로 사용할 새로운 page를 할당 하고, 할당된 page의 page ID를 반환함
  - 파라미터
    - ObjectID \*catObjForFile  
(IN) page를 할당할 B+ tree 색인 file 및 색인된 데이터 file에 대한 정보 (sm\_CatOverlayForSysTables) 가 저장된 object의 OID
    - PageID \*nearPid  
(IN) 할당할 page의 page ID 또는 할당할 page가 disk 상에서 물리적으로 인접해야 하는 page의 page ID
    - PageID \*newPid  
(OUT) 할당된 page의 page ID
  - 반환값
    - Four 에러코드

## — 예

```
Four EduBtM_CreateIndex(  
    ObjectID *catObjForFile,  
    /* IN: ID of the object that contains the catalog information */  
    PageID *rootPid) /* OUT: ID of the root page of the newly created B+tree */  
{  
    sm_CatOverlayForBtree *catEntry;  
    /* pointer to the B+tree file catalog information */  
    PhysicalFileID pFid; /* ID of the first page in the file */  
    ...  
    MAKE_PHYSICALFILEID(pFid, catEntry->fid.volNo, catEntry->firstPage);  
    /* Allocate a new page to be used as a B+ tree index page */  
    e = btm_AllocPage(catObjForFile, (PageID *)&pFid, rootPid);  
    if (e < 0) ERR(e);  
    ...  
}
```

- btm\_ObjectIdComp()
  - 파라미터로 주어진 두 object ID (OID) 의 대소를 비교하고, 비교 결과를 반환함
  - 파라미터
    - ObjectID \*firstOid  
(IN) 비교할 첫 번째 object ID
    - ObjectID \*secondOid  
(IN) 비교할 두 번째 object ID
  - 반환값
    - Four EQUAL, GREATER, 또는 LESS

## — 예

```
Four edubtm_DeleteLeaf(...
    ObjectID *oid, /* IN: ID of the object to be deleted */
    ...)
{
    ObjectID tOid;      /* ID of an object */
    ...
    /* Compare two object IDs */ if(edubtm_ObjectIdComp(oid, &tOid) == EQUAL) {...}
    ...
}
```

- btm\_root\_delete()
  - B+ tree 색인의 root page에서 발생한 underflow를 처리함
  - 파라미터
    - PhysicalFileID \*pFid  
(IN) B+ tree 색인 file의 file ID (= B+ tree 색인 file의 첫 번째 page의 page ID)
    - PageID \*rootPid  
(IN) B+ tree 색인의 root page의 page 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



## — 예

```
Four EduBtM_DeleteObject(...
    PageID *root, /* IN: ID of the root page */
    ...
    Pool *dlPool, /* INOUT: pool of the elements of the dealloc list */
    DeallocListElem *dlHead) /* INOUT: head of the dealloc list */
{
    Boolean lf; /* TRUE if a page is not half full */
    PhysicalFileID pFid; /* ID of the index file containing the B+ tree index */
    ...
    /* Handle underflow that has occurred in the root page of a B+ tree index */
    if(lf) { // if underflow has occurred in the root page
        e = btm_root_delete(&pFid, root, dlPool, dlHead); if (e < 0) ERR(e);
    }
    ...
}
```

- btm\_Underflow()
  - B+ tree 색인의 under flow가 발생한 page에 대해, 해당 page를 sibling page와 merge 또는 redistribute 함
    - 두 page가 merge 되는 경우, 부모 page에서 두 page와 관계된 index entry가 삭제됨에 따라 부모 page에서 underflow가 발생할 수 있음
    - 두 page가 redistribute 되는 경우, 부모 page에서 두 page와 관계된 index entry가 교체됨 (기존 index entry가 삭제되고 새로운 index entry가 삽입됨)에 따라 부모 page에서 overflow가 발생할 수 있음
  - Internal node일 경우, under flow가 발생한 page가 p0가 가리키는 page면 edubtm\_BinarySearchInternal() 함수에 의해 반환된 값인 -1을 slotNo에 input으로 넣는다.
  - 파라미터
    - PhysicalFileID \*pFid  
(IN) B+ tree 색인 file의 file ID (= B+ tree 색인 file의 첫 번째 page의 page ID)
    - BtreePage \*rpage  
(IN) under flow가 발생한 page의 부모 page가 저장된 buffer element에 대한 포인터
    - PageID \*child  
(IN) under flow가 발생한 page의 page ID
    - Two slotNo  
(IN) under flow가 발생한 page를 가리키는 index entry의 offset이 저장된 slot의 slot 번호
    - Boolean \*f  
(OUT) 부모 page에서 underflow가 발생했음을 나타내는 flag
    - Boolean \*h  
(OUT) 부모 page에서 overflow가 발생했음을 나타내는 flag
    - InternalItem \*item  
(OUT) 부모 page의 overflow로 인해 삽입되지 못한 internal index entry
    - 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

## — 예

```

Four edubtm_Delete(...)
    PageID *root, /* IN: ID of the root page */
    Boolean *f, /* OUT: whether the root page is half full */
    ...
    Pool *dlPool, /* INOUT: pool of the elements of the dealloc list */
    DeallocListElem *dlHead) /* INOUT: head of the dealloc list */
{
    Boolean lf; /* TRUE if a page is not half full */
    Boolean lh; /* TRUE if a page is splitted */ Two idx; /* slot number */
    PageID child; /* ID of the child page */
    BtreePage *rpage; /* pointer to the root page */
    InternalItem litem; /* Internal index entry */
    PhysicalFileID pFid; /* ID of the index file containing the B+ tree index */
    ...
    /* Merge or redistribute the page with the sibling page */
    else if (lf) { // if underflow occurs
        e = btm_Underflow(&pFid, rpage, &child, idx, f, &lh, &litem, dlPool, dlHead);
        if (e < 0) ERRB1(e, root, 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(root == NULL) ERR(eBADPARAMETER_BTМ)
```

- ERRB1(e, pid, t)

- Error code *e*를 반환하기 전에 파라미터로 주어진 *pid*에 대응하는 page를 unfix 하는 것을 제외하고 ERR와 동일함

- 사용예

- ```
if(e < 0) ERRB1(e, &newPid, PAGE_BUF)
```

- Error code

- ```
$(EduBtM_HOME_DIR)/Header/EduBtM_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 \$(EduBtM\_HOME\_DIR)/Header 디렉토리의 헤더 파일들에 저장된 각종 macro들을 활용 가능함
- Test 폴더의 autograder.sh를 실행함
  - •cd test; bash autograder.sh
- 오토그레이더는 총 30개의 YCSB 워크로드를 실행함
  - 15개는 에러 테스트에 사용됨 (개당 6점, 기본 10점)
  - 나머지 15개는 성능 측정에 사용됨 (채점에 반영되지 않음)

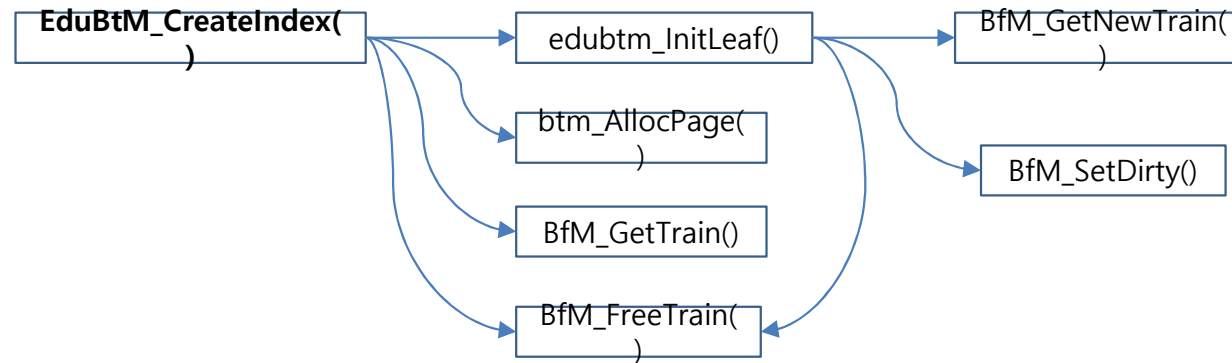
- ❖ 일부 기능만을 구현하여 테스트 하는 방법

테스트 모듈 소스 코드 내에서, 구현하지 않은 function을 호출하는 대신 그에 해당하는 successful default solution function (function 명에서 "Edu" keyword가 제거된 형태) 을 호출하도록 수정함

예) EduBtM\_CreateIndex() function의 successful default solution function은 BtM\_CreateIndex()  
임

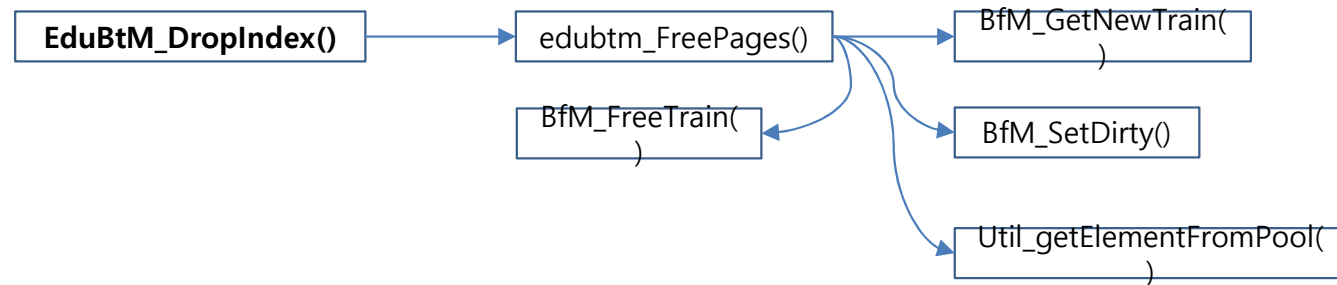
# Appendix

# Function Call Graph: EduBtM\_CreateIndex()

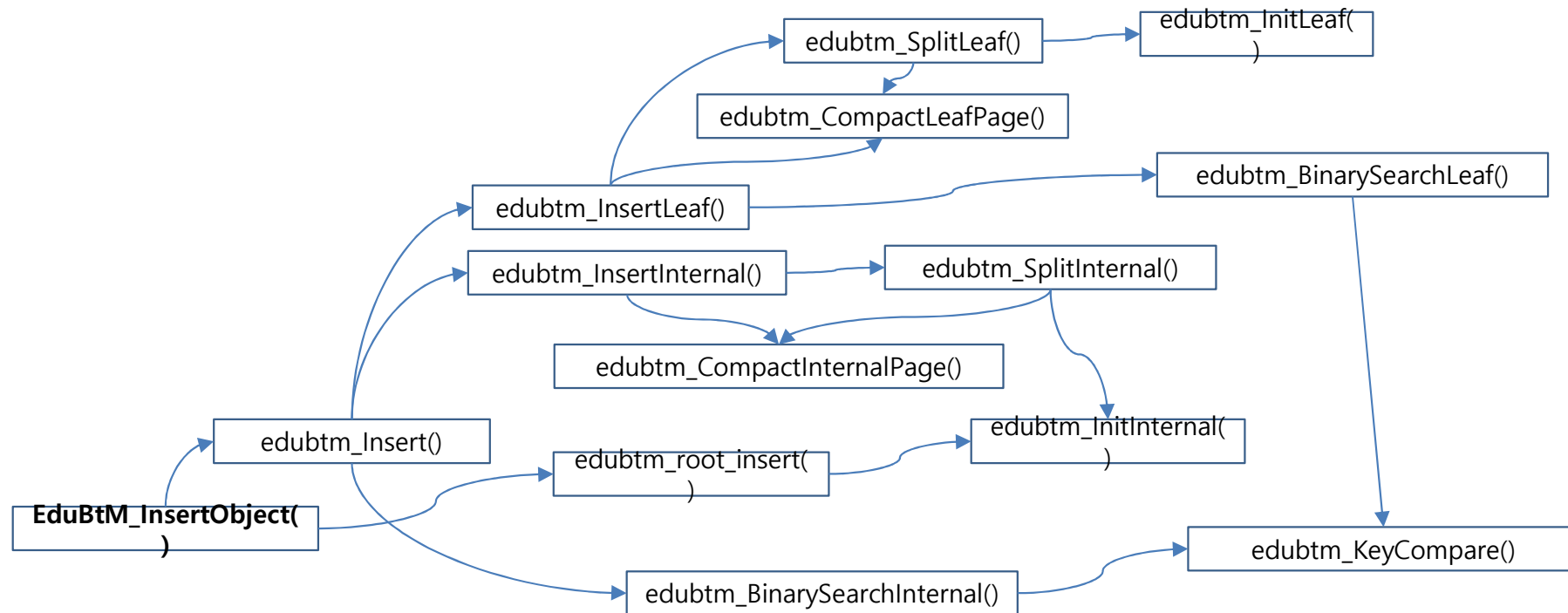




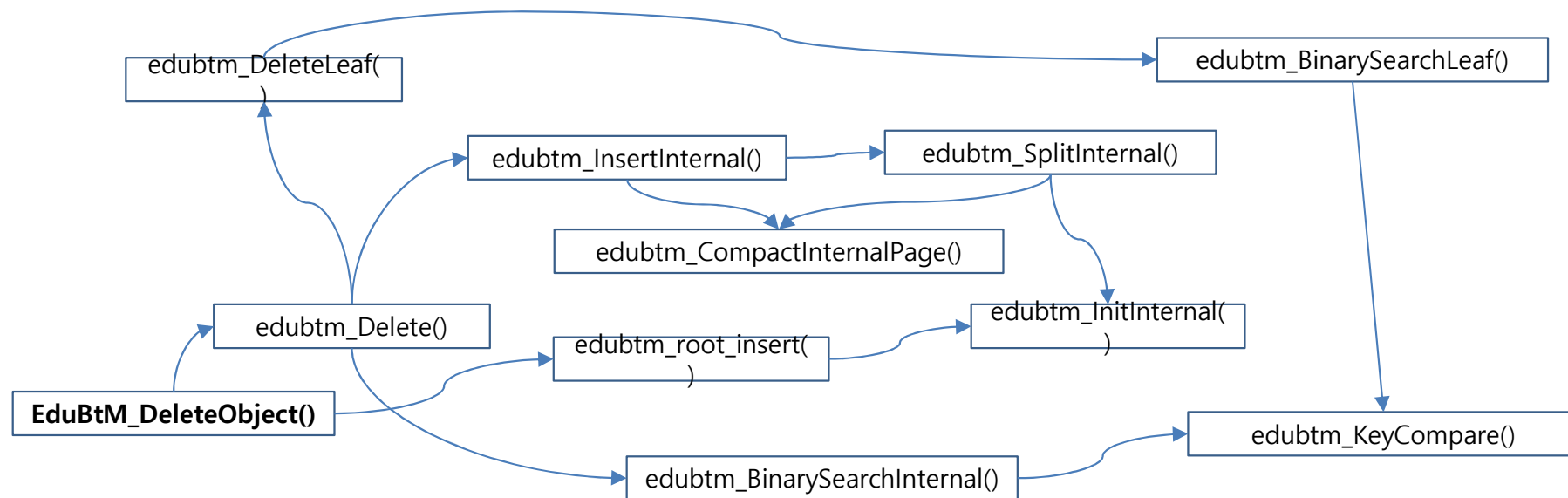
# Function Call Graph: EduBtM\_DropIndex()



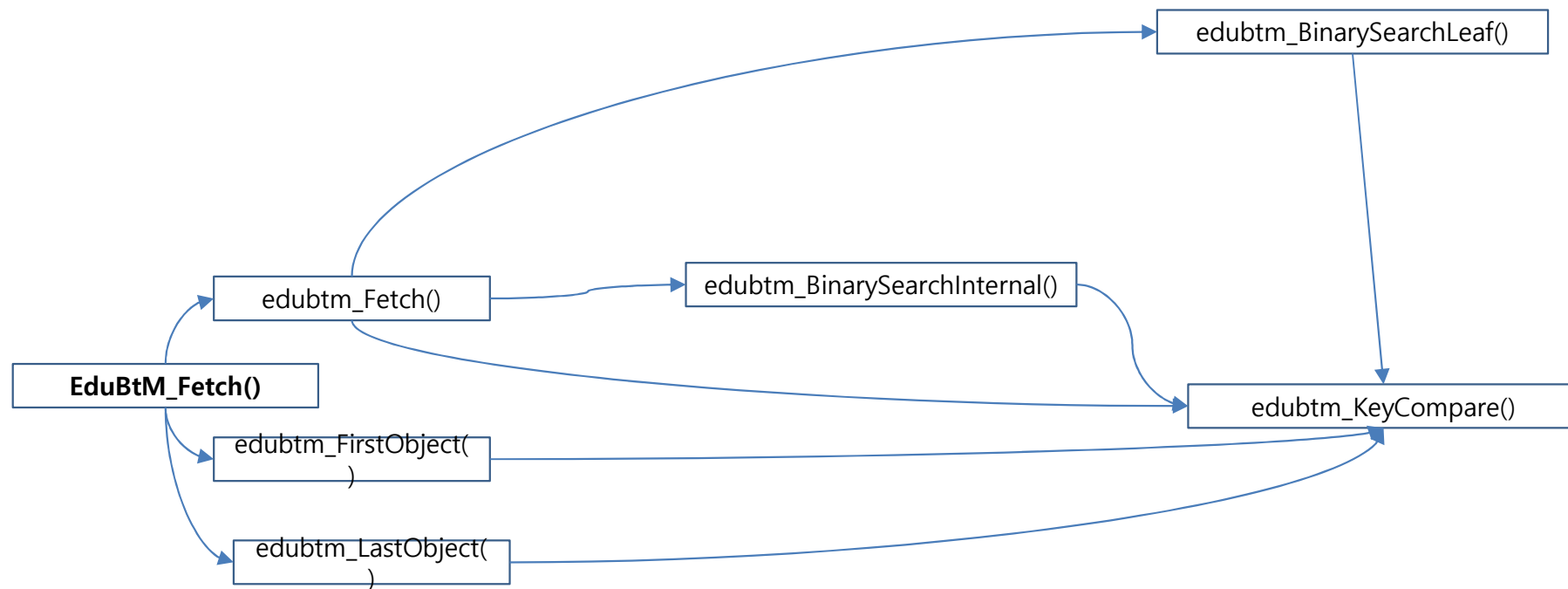
# Function Call Graph: EduBtM\_InsertObject()



# Function Call Graph: EduBtM\_DeleteObject()



# Function Call Graph: EduBtM\_Fetch()



# Function Call Graph: EduBtM\_FetchNext()

