

오디세우스/EduCOSMOS Project #1: EduBfM Project Manual

Version 1.1

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.

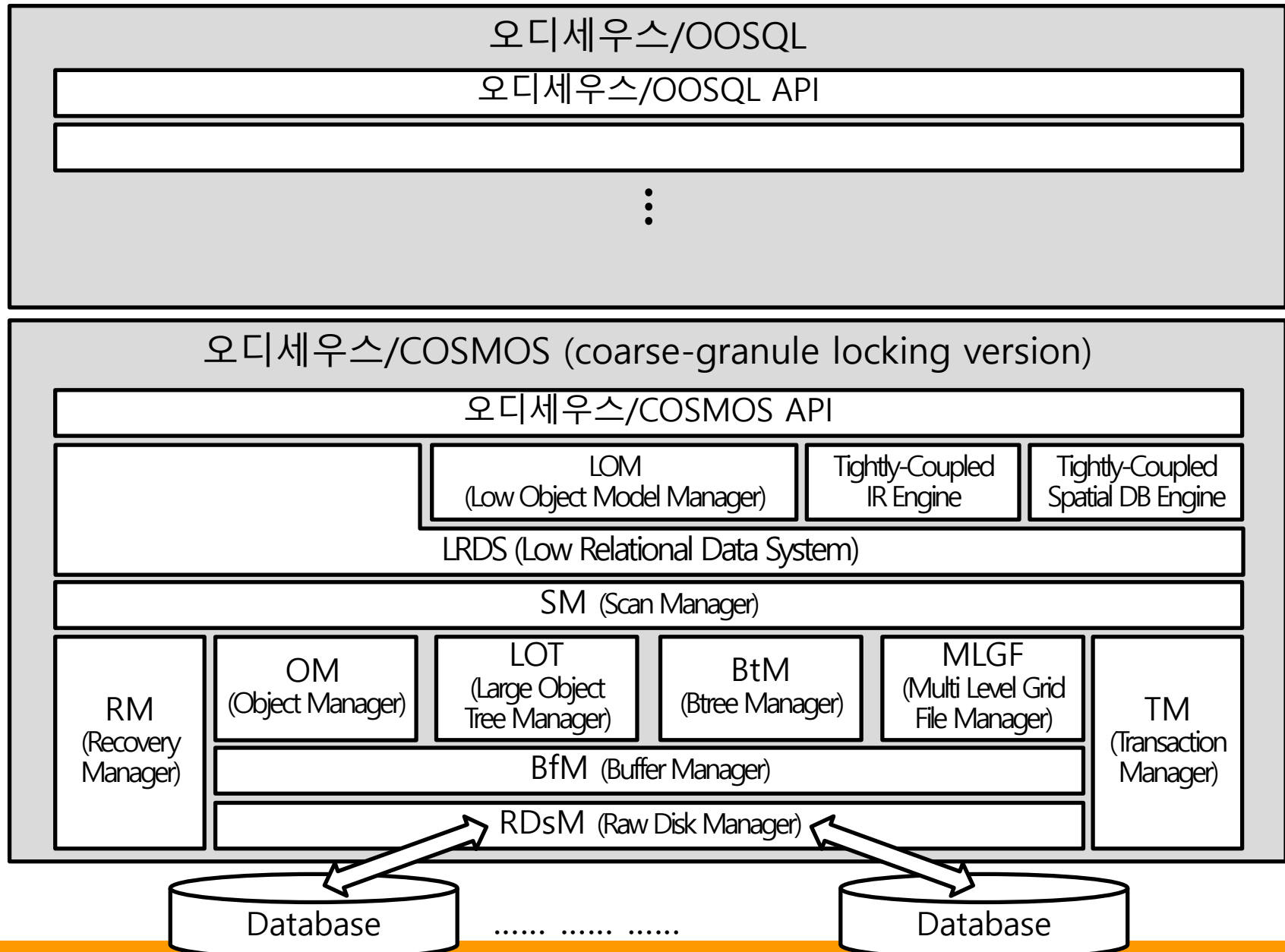
목차

- 소개
 - 오디세우스/COSMOS
 - 오디세우스/EduCOSMOS Project
- EduBfM 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



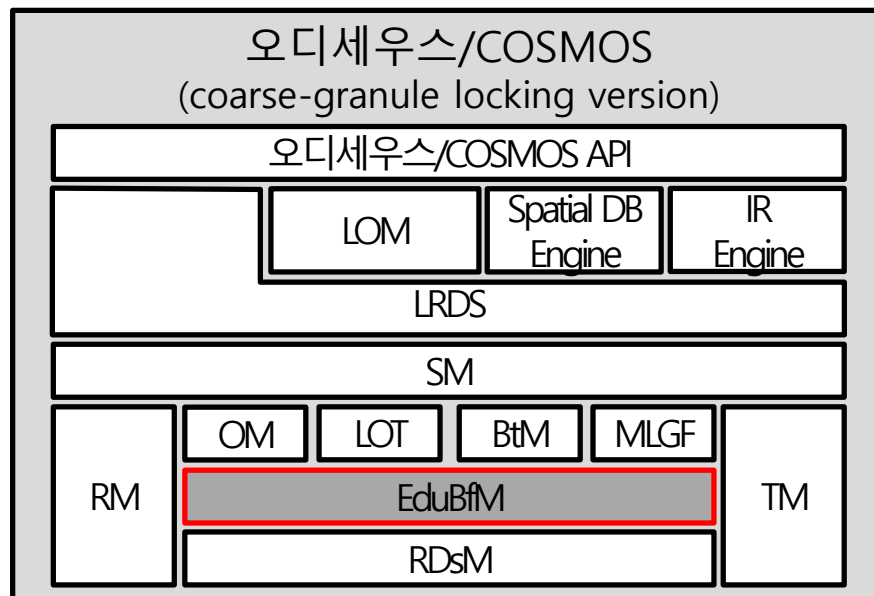
오디세우스/EduCOSMOS Project

- 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

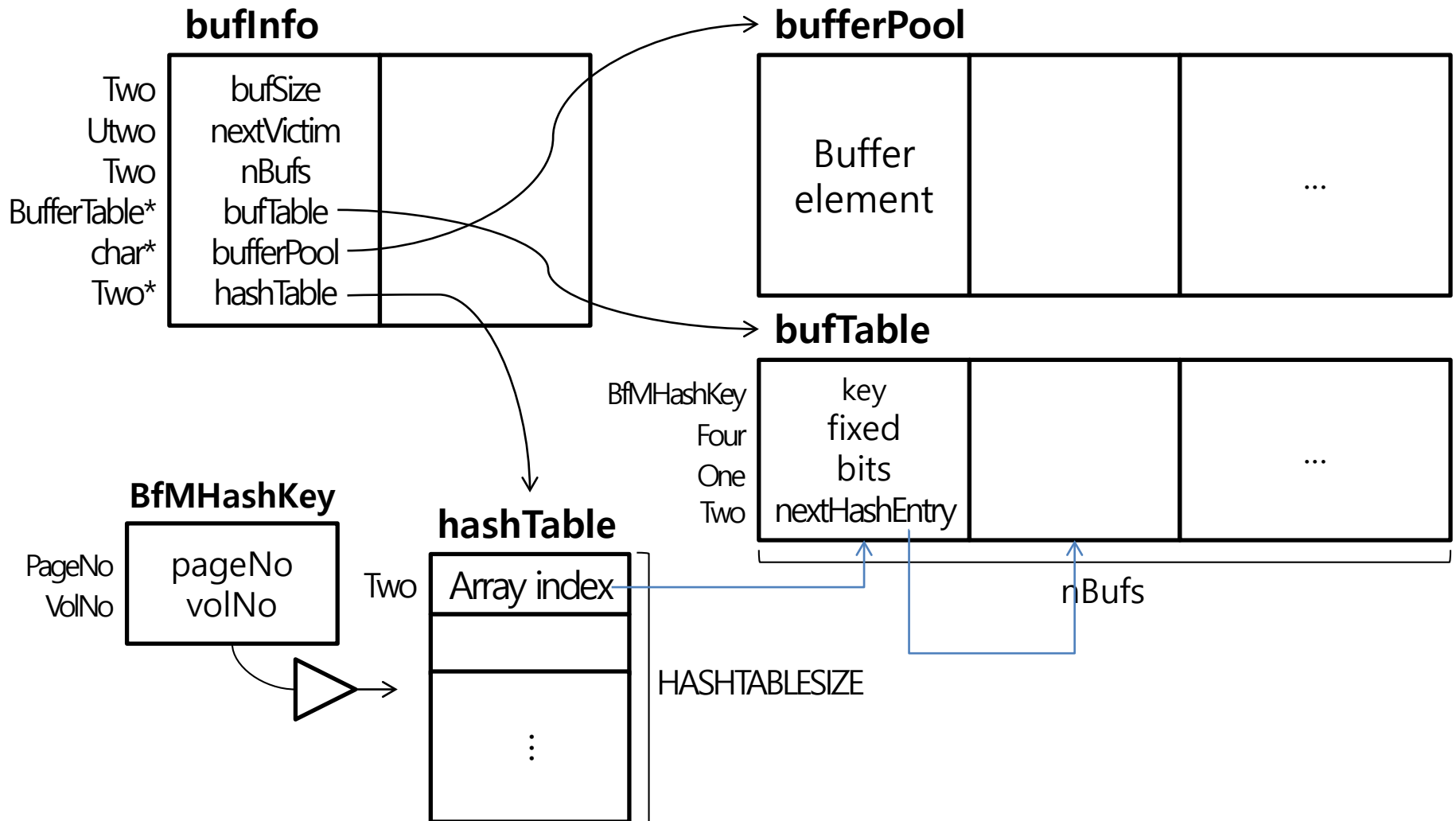
EduBfM Project

- 목표

- Disk 상의 page 또는 train을 main memory 상에 유지하는 buffer manager에 대한 데이터 구조 및 연산들을 구현함
 - Train: page의 데이터 영역보다 큰 large object를 저장하기 위한 page들의 집합
- EduBfM에서는 오디세우스/COSMOS BfM의 기능들 중 극히 제한된 일부 기능들만을 구현함



데이터 구조



bufInfo

- 개요

- Buffer pool 및 관련 정보를 유지하기 위한 데이터 구조
- 두 가지 종류의 buffer pool이 존재하며, 각각 별도의 *bufInfo*를 유지함
 - PAGE_BUF: page의 전체 데이터 영역보다 작은 small object들, index entry들 등이 저장되는 page들을 저장하기 위한 page buffer pool
 - LOT_LEAF_BUF: page의 데이터 영역보다 큰 large object를 나타내는 구조인 Large Object Tree의 leaf node (현재, 4개의 page들로 구성됨) 들을 저장하기 위한 train buffer pool

- 구성

- bufSize
 - Buffer pool을 구성하는 buffer element의 크기 (단위: # of pages)
 - PAGE_BUF: 1
 - LOT_LEAF_BUF: 4 (조절 가능함)

- nextVictim
 - Buffer replacement algorithm이 buffer element의 교체 여부 판별을 위해 방문할 다음 buffer element의 array index
- nBufs
 - Buffer pool을 구성하는 buffer element의 개수
- bufTable
 - Buffer pool을 구성하는 각 buffer element에 대한 정보를 저장하는 table
- bufferPool
 - Buffer pool (disk로부터 읽어온 page들/train들을 main memory 상에 유지하기 위한 buffer element들의 집합)
- hashTable
 - Buffer pool에 존재하는 page들/train들에 대한 효율적인 검색을 지원하기 위한 hash table

bufTable

- 개요

- *bufferPool*을 구성하는 각 buffer element에 저장된 page/train에 대한 정보를 저장하기 위한 데이터 구조
- *bufTable*의 n 번째 element는 *bufferPool*의 n 번째 buffer element에 저장된 page/train에 대한 정보를 저장함

- 구성

- key
 - Buffer element에 저장된 page/train의 hash key (= page의 ID 또는 train의 첫 번째 page의 ID)
 - Page의 ID는 page 번호 및 volume 번호로 구성됨
- fixed
 - Buffer element에 저장된 page/train을 fix (access) 하고 있는 transaction들의 수

– bits

- buffer element의 상태를 나타내는 bit들의 집합

- 첫 번째 bit (DIRTY): buffer element에 저장된 page/train이 수정되었음을 나타내는 bit
- 세 번째 bit (REFER): buffer replacement algorithm에 의한 방문 여부를 나타내는 bit
- 이외의 bit들은 EduBfM에서는 사용하지 않음 (EduBfM function 구현시 이를 무시해도 됨)

– nextHashEntry

- 동일한 hash value를 갖는 다음 page/train이 저장된 buffer element의 array index

BfMHashKey

- 개요
 - Page/train의 hash key를 저장하기 위한 데이터 구조
- 구성
 - pageNo
 - Page 번호 또는 train의 첫 번째 page 번호
 - 한 volume 내에서 page별로 unique한 번호임
 - volNo
 - Page/train이 저장된 disk volume의 volume 번호
 - 한 시스템 내에서 volume별로 unique한 번호임

※ Hash value는 hash key를 input으로 하는 hash function의 output임

$$\text{Hash value} = (\text{pageNo} + \text{volNo}) \% \text{HASHTABLESIZE}$$

hashTable

- 개요

- Page/train이 저장된 buffer element의 array index를 저장하기 위한 table로서, buffer element의 *bufTable*에 대한 array index를 저장하는 *hashTable* entry들로 구성됨
- Buffer element에 저장된 page/train의 hash value를 이용하여 array index를 *hashTable* entry에 저장함
 - Hash value가 n 인 page/train 중 가장 최근에 disk로부터 읽어온 page/train이 저장된 buffer element의 array index는 *hashTable*의 n 번째 entry에 저장됨
 - Hash value가 동일한 다른 page/train들이 저장된 buffer element들의 array index들은 *bufTable*의 *nextHashEntry* 변수를 통해 linked list 형태로 연결되어 유지됨
- Array index가 저장되지 않은 *hashTable* entry에는 *NIL*(-1)값이 저장됨

관련 연산

- Page/train을 fix
 - Page/train을 access하기 위해 해당 page/train을 *bufferPool*에 fix 함
 - 모든 transaction들은 page/train을 access하기 전에 해당 page/train을 *bufferPool*에 fix 해야 함
 - *fixed* 변수 값을 1 증가시킴
- Page/train을 unfix
 - Page/train을 *bufferPool*에서 unfix 함
 - 모든 transaction들은 page/train access를 마치고 해당 page/train을 *bufferPool*에서 unfix 해야 함
 - *fixed* 변수 값을 1 감소시킴
- Dirty bit를 set
 - *bufferPool*에 저장된 page/train이 수정되었음을 표시하기 위해 DIRTY bit를 1로 set 함
- *bufferPool*의 page/train들을 flush
 - *bufferPool*에 존재하는 page/train들 중 수정된 page/train들을 disk에 기록함
- *bufferPool*의 page/train들을 discard
 - *bufferPool*에 존재하는 page/train들을 *bufferPool*에서 삭제함

구현할 API Function 들

- EduBfM_GetTrain()
- EduBfM_FreeTrain()
- EduBfM_SetDirty()
- EduBfM_FlushAll()
- EduBfM_DiscardAll()

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

(※ API: Application Programming Interface)

EduBfM_GetTrain()

- 파일: EduBfM_GetTrain.c
- 설명
 - Page/train을 *bufferPool*에 fix 하고, page/train이 저장된 buffer element에 대한 포인터를 반환함
 - Fix 할 page/train의 hash value를 이용하여, 해당 page/train이 저장된 buffer element의 array index를 *hashTable*에서 검색함
 - Fix 할 page/train이 *bufferPool*에 존재하지 않는 경우,
 - *bufferPool*에서 page/train을 저장할 buffer element 한 개를 할당 받음
 - Page/train을 disk로부터 읽어와서 할당 받은 buffer element에 저장함
 - 할당 받은 buffer element에 대응하는 *bufTable* element를 갱신함
 - » key: fix 할 page/train의 hash key를 저장함
 - » fixed: 1로 설정함

- » *bits*: REFER bit를 1로 set함
 - DIRTY bit는 EduBfM_SetDirty(), EduBfM_DiscardAll(), edubfm_AllocTrain(), edubfm_FlushTrain()에 의해 manage되므로 갱신할 필요 없음
 - » *nextHashEntry*: edubfm_Insert(), edubfm_Delete()에 의해 manage되므로 갱신할 필요 없음
- 할당 받은 buffer element의 array index를 *hashTable*에 삽입함
- 할당 받은 buffer element에 대한 포인터를 반환함
- Fix 할 page/train이 *bufferPool*에 존재하는 경우,
 - 해당 page/train이 저장된 buffer element에 대응하는 *bufTable* element를 갱신함
 - » *fixed*: 1 증가시킴
 - » *bits*: REFER bit를 1로 set함
 - 해당 buffer element에 대한 포인터를 반환함

- 파라미터
 - PageID/TrainID *trainId
(IN) fix 할 page의 ID 또는 train의 첫 번째 page의 ID
 - char **retbuf
(OUT) fix 된 page/train이 저장될 buffer element에 대한 포인터
 - Four type
(IN) *bufferPool*의 종류
- 반환값
 - Four 에러코드
 - eNOERROR: 에러 없음
 - 관련 함수에서 반환되는 에러코드
- 관련 함수
edubfm_AllocTrain(), edubfm_Insert(), edubfm_LookUp(),
edubfm_ReadTrain()

EduBfM_FreeTrain()

- 파일: EduBfM_FreeTrain.c
- 설명
 - Page/train을 *bufferPool*에서 unfix 함
 - Unfix 할 page/train의 hash value를 이용하여, 해당 page/train이 저장된 buffer element의 array index를 *hashTable*에서 검색함
 - 해당 buffer element에 대한 *fixed* 변수 값을 1 감소시킴
 - *fixed* 변수의 값이 0보다 작아질 경우,
 - » 경고 메시지 ("Warning: Fixed counter is less than 0!!!") 를 화면에 출력함
 - » *fixed* 변수의 값을 0으로 설정함

- 파라미터
 - PageID/TrainID *trainId
(IN) unfix 할 page의 ID 또는 train의 첫 번째 page의 ID
 - Four type
(IN) *bufferPool*의 종류
- 반환값
 - Four 에러코드
 - eNOERROR: 에러 없음
 - eNOTFOUND_BFM: page/train이 *bufferPool*에 존재하지 않음
- 관련 함수
edubfm_LookUp()

EduBfM_SetDirty()

- 파일: EduBfM_SetDirty.c
- 설명
 - *bufferPool*에 저장된 page/train이 수정되었음을 표시하기 위해 DIRTY bit를 1로 set함
 - 수정된 page/train의 hash value를 이용하여, 해당 해당 page/train이 저장된 buffer element의 array index를 *hashTable*에서 검색함
 - 해당 buffer element에 대한 DIRTY bit를 1로 set함

- 파라미터

- PageID/TrainID *trainId
(IN) 수정된 page의 ID 또는 train의 첫 번째 page의 ID
- Four type
(IN) *bufferPool*의 종류

- 반환값

- Four 에러코드
 - eNOERROR: 에러 없음
 - eNOTFOUND_BFM: page/train이 *bufferPool*에 존재하지 않음

- 관련 함수

edubfm_LookUp()

EduBfM_FlushAll()

- 파일: EduBfM_FlushAll.c
- 설명
 - 각 *bufferPool*에 존재하는 page/train들 중 수정된 page/train들을 disk에 기록함
 - DIRTY bit가 1로 set 된 buffer element들에 저장된 각 page/train에 대해, `edubfm_FlushTrain()`을 호출하여 해당 page/train을 disk에 기록함

- 파라미터 없음
- 반환값
 - Four 에러코드
 - eNOERROR: 에러 없음
 - 관련 함수에서 반환되는 에러코드
- 관련 함수
edubfm_FlushTrain()

EduBfM_DiscardAll()

- 파일: EduBfM_DiscardAll.c
- 설명
 - 각 *bufferPool*에 존재하는 page/train들을 disk에 기록하지 않고 *bufferPool*에서 삭제함
 - 각 *bufTable*의 모든 element들을 초기화함
 - *key.pageNo*를 *NIL*(-1)로 설정함
 - *fixed*: EduBfM_GetTrain(), EduBfM_FreeTrain()에 의해 manage되므로 초기화할 필요 없음
 - *bits*: 모든 bit를 reset함
 - *nextHashEntry*: edubfm_Insert(),edubfm_Delete()에 의해 manage되므로 초기화할 필요 없음
 - 각 *hashTable*에 저장된 모든 entry (즉, array index) 들을 삭제함

- 파라미터 없음
- 반환값
 - Four 에러코드
 - eNOERROR: 에러 없음
 - 관련 함수에서 반환되는 에러코드
- 관련 함수
edubfm_DeleteAll()

구현할 Internal Function 들

- edubfm_ReadTrain()
- edubfm_AllocTrain()
- edubfm_Insert()
- edubfm_Delete()
- edubfm_Deleteall()
- edubfm_LookUp()
- edubfm_FlushTrain()

edubfm_ReadTrain()

- 파일: edubfm_ReadTrain.c
- 설명
 - Page/train을 disk로부터 읽어와서 buffer element에 저장하고, 해당 buffer element에 대한 포인터를 반환함

- 파라미터
 - PageID/TrainID *trainId
(IN) 읽어올 page의 ID 또는 train의 첫 번째 page의 ID
 - char *aTrain
(OUT) 읽어온 page/train이 저장될 buffer element에 대한 포인터
 - Four type
(IN) *bufferPool*의 종류
- 반환값
 - Four 에러코드
 - eNOERROR: 에러 없음
 - 관련 함수에서 반환되는 에러코드
- 관련 함수
RDsM_ReadTrain() (p.44~45 참고)

edubfm_AllocTrain()

- 파일: edubfm_AllocTrain.c
- 설명
 - *bufferPool*에서 *page/train*을 저장하기 위한 buffer element를 한 개 할당 받고, 해당 buffer element의 array index를 반환함
 - Second chance buffer replacement algorithm을 사용하여, 할당 받을 buffer element를 선정함
 - 할당 대상 선정을 위해 대응하는 *fixed* 변수 값이 0인 buffer element들을 순차적으로 방문함
 - » 방문 순서:
 - 1) Array index가 *bufInfo.nextVictim*인 buffer element
 - 2) Array index가 $(\text{bufInfo.nextVictim}+1)\% \text{bufInfo.nBufs}$ 인 buffer element
 - ...
 - n) Array index가 $(\text{bufInfo.nextVictim}+n-1)\% \text{bufInfo.nBufs}$ 인 buffer element

- 선정된 buffer element와 관련된 데이터 구조를 초기화함
 - 선정된 buffer element에 저장되어 있던 page/train이 수정된 경우, 기존 buffer element의 내용을 disk로 flush함
 - 선정된 buffer element에 대응하는 *bufTable* element를 초기화함
 - » *key*: *EduBfM_GetTrain()*, *EduBfM_DiscardAll()* 에 의해 manage되므로 초기화할 필요 없음
 - » *fixed*: *EduBfM_GetTrain()*, *EduBfM_FreeTrain()*에 의해 manage되므로 초기화할 필요 없음
 - » *bits*: 모든 bit를 reset함
 - » *nextHashEntry*: *edubfm_Insert()*, *edubfm_Delete()*에 의해 manage되므로 초기화할 필요 없음
 - *bufInfo.nextVictim*의 값을 ((선정된 buffer element의 array index + 1) % *bufInfo.nBufs*)로 설정함
 - 선정된 buffer element의 array index (*hashTable* entry) 를 *hashTable* (linked list)에서 삭제함
- 선정된 buffer element의 array index를 반환함

- 파라미터
 - Four type
(IN) *bufferPool*의 종류
- 반환값
 - Four 할당 받은 buffer element의 array index
또는 에러코드
 - eNOUNFIXEDBUF_BFM: unfix된 (즉, *fixed* 변수 값이 0인) buffer element가 없음
 - 관련 함수에서 반환되는 에러코드
- 관련 함수
 - edubfm_Delete(), edubfm_FlushTrain()

edubfm_Insert()

- 파일: edubfm_Hash.c
- 설명
 - *hashTable*에 buffer element의 array index를 삽입함
 - 해당 buffer element에 저장된 page/train의 hash value를 이용하여, *hashTable*에서 해당 array index를 삽입할 위치를 결정함
 - Hash value가 n 인 경우, *hashTable*의 n 번째 entry에 삽입함
 - Collision이 발생하지 않은 경우, 해당 array index를 결정된 위치에 삽입함
 - Collision이 발생한 경우, chaining 방법을 사용하여 이를 처리함
 - 해당 buffer element에 대한 *nextHashEntry* 변수에 기존 *hashTable* entry (array index) 를 저장함
 - 새로운 array index를 결정된 위치에 삽입함
 - ⇒ 동일한 hash value를 갖는 page/train들이 저장된 buffer element들의 array index들이 linked list 형태로 연결되어 유지됨

- 파라미터
 - BfMHashKey *key
(IN) buffer element에 저장된 page/train의 hash key
 - Two index
(IN) 삽입할 buffer element의 array index
 - Four type
(IN) *bufferPool*의 종류
- 반환값
 - Four 에러코드
 - eNOERROR: 에러 없음
- 관련 함수 없음

edubfm_Delete()

- 파일: edubfm_Hash.c
- 설명
 - *hashTable*에서 buffer element의 array index를 삭제함
 - 해당 buffer element에 저장된 page/train의 hash value를 이용하여, 삭제할 buffer element의 array index를 *hashTable*에서 검색함
 - 검색된 entry (array index) 를 *hashTable*에서 삭제함
 - 동일한 hash value를 갖는 page/train들이 저장된 buffer element들의 array index들간의 linked list 구조가 유지되도록 해당 array index를 삭제함

- 파라미터
 - BfMHashKey *key
(IN) buffer element에 저장된 page/train의 hash key
 - Four type
(IN) *bufferPool*의 종류
- 반환값
 - Four 에러코드
 - eNOERROR: 에러 없음
 - eNOTFOUND_BFM: *key*가 *hashTable*에 존재하지 않음
- 관련 함수 없음

edubfm_DeleteAll()

- 파일: edubfm_Hash.c
- 설명
 - 각 *hashTable*에서 모든 entry (buffer element의 array index) 들을 삭제함
 - *hashTable* entry들의 값을 *NIL*(-1)로 설정함

- 파라미터 없음
- 반환값
 - Four 에러코드
 - eNOERROR: 에러 없음
- 관련 함수 없음

edubfm_LookUp()

- 파일: edubfm_Hash.c
- 설명
 - 파라미터로 주어진 hash key (BfMHashKey) 를 갖는 page/train 이 저장된 buffer element의 array index를 *hashTable*에서 검색하여 반환함
 - 해당 hash key를 갖는 page/train이 저장된 buffer element의 array index를 *hashTable*에서 검색함
 - 해당 hash key (*key*) 를 사용하여 hash value (*hashValue*) 를 계산함
 - » $hashValue = (key->volNo + key->pageNo) \% HASHTABLESIZE$
 - *hashValue* 번째 *hashTable* entry로 저장된 array index가 가리키는 buffer element를 시작으로 하여 *nextHashEntry*로 연결된 buffer element들을 순차적으로 방문하여 주어진 hash key를 갖는 page/train이 저장된 buffer element를 찾음
 - 검색된 buffer element의 array index를 반환함

- 파라미터
 - BfMHashKey *key
(IN) 검색에 사용할 hash key
 - Four type
(IN) *bufferPool*의 종류
- 반환값
 - Four 검색된 buffer element의 array index
또는, 에러코드
 - NOTFOUND_IN_HTABLE: *key*가 *hashTable*에 존재하지 않음
- 관련 API 및 함수 없음

edubfm_FlushTrain()

- 파일: edubfm_FlushTrain.c
- 설명
 - 수정된 page/train을 disk에 기록함
 - Flush 할 page/train의 hash value를 이용하여, 해당 page/train이 저장된 buffer element의 array index를 *hashTable*에서 검색함
 - 해당 buffer element에 대한 DIRTY bit가 1로 set 된 경우, 해당 page/train을 disk에 기록함
 - 해당 DIRTY bit를 reset 함

- 파라미터
 - PageID/TrainID *trainId
(IN) flush 할 page의 ID 또는 train의 첫 번째 page의 ID
 - Four type
(IN) *bufferPool*의 종류

- 반환값
 - Four 에러코드
 - eNOERROR: 에러 없음
 - eNOTFOUND_BFM: page/train이 *bufferPool*에 존재하지 않음
 - 관련 함수에서 반환되는 에러코드

- 관련 함수
edubfm_LookUp(), RDsM_WriteTrain() (p.46~47 참고)

제공되는 Function 들

- RDsM_ReadTrain()
 - Page/train을 디스크로부터 읽어옴
 - 파라미터
 - PageID/TrainID *trainId
(IN) 읽어올 page의 ID 또는 train의 첫 번째 page의 ID
 - char *bufPtr
(OUT) 읽어온 page/train이 저장될 buffer element에 대한 포인터
 - Two sizeofTrain
(IN) 읽어올 train의 크기 (단위: # of pages)
(※ page를 읽어오기 위해서는, *sizeofTrain*을 1로 설정함)
 - 반환값
 - Four 에러코드

– 예

```
Four edubfm_ReadTrain(  
    TrainID *trainId, /* IN: ID of the page to be read */  
    char *aTrain, /* OUT: pointer to the buffer */  
    Four type) /* IN: buffer type */  
{  
    ...  
    /* Read the page from the disk */  
    e = RDsM_ReadTrain(trainId, aTrain, BI_BUFSIZE(type));  
    // RDsM_ReadTrain() 호출 결과 읽어온 page/train은 aTrain 변수가 가리키는 곳에 저장됨  
    if( e < 0 ) ERR( e );  
    ...  
}
```

- RDSM_WriteTrain()
 - Page/train을 디스크에 기록함
 - 파라미터
 - char *bufPtr
(IN) 기록할 page/train이 저장된 buffer element에 대한 포인터
 - PageID *trainId
(IN) 기록할 page의 ID 또는 train의 첫 번째 page의 ID
 - Two sizeOfTrain
(IN) 기록할 train의 크기 (단위: # of pages)
(※ page를 기록하기 위해서는, *sizeOfTrain*을 1로 설정함)
 - 반환값
 - Four 에러코드

— 예

```
Four edubfm_FlushTrain(  
    TrainID *trainId, /* IN: ID of the page to be flushed */  
    Four type) /* IN: buffer type */  
{  
    Four index; /* array index of the buffer element that contains the page */  
    ...  
    /* Write the page into the disk */  
    e = RDsM_WriteTrain(BI_BUFFER(type, index), trainId, BI_BUFSIZE(type));  
    if( e < 0 ) ERR( e );  
    ...  
}
```

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(retBuf == NULL) ERR(eBADBUFFER_BFM)`

- Error code

`$(EduBfM_HOME_DIR)/Header/EduBfM_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 prototype들로 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
 - 기타 파일
 - Volume 파일 (.vol 파일)
테스팅 중 사용되는 데이터가 저장되는 파일

- Project 수행 방법

- Skeleton 파일 내의 function들을 구현함
 - 구현시 \$(EduBfM_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

- ❖ \$(EduBfM_HOME_DIR)/Header/EduBfM_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

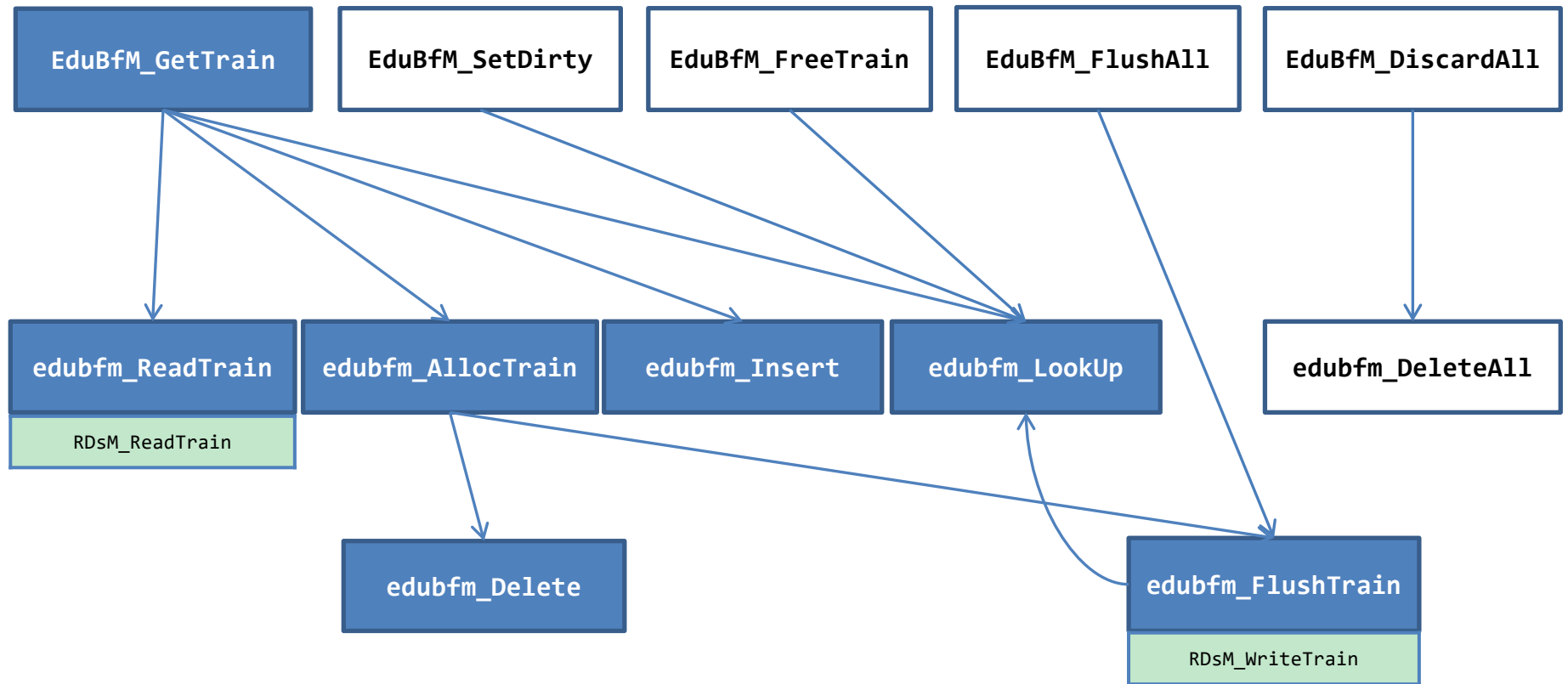
- ❖ API function에서 호출하는 일부 internal function을 구현하지 않고 해당 API function을 구현하는 방법

- ❖ 대응하는 default solution function (internal function 명에서 "edu" keyword가 제거된 형태)을 사용함
 - ❖ 예: edubfm_ReadTrain()의 successful default solution function은 bfm_ReadTrain() 임

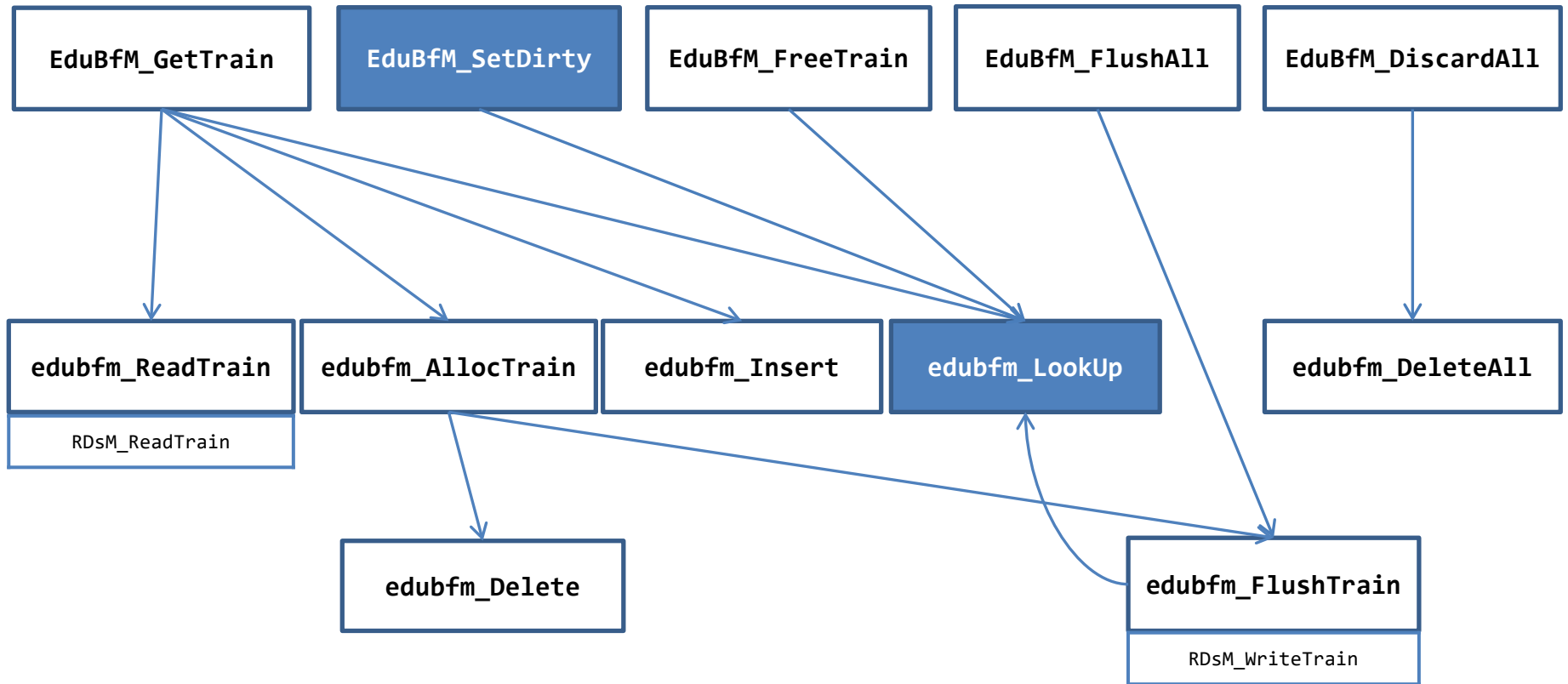
Appendix

Function Call Graph

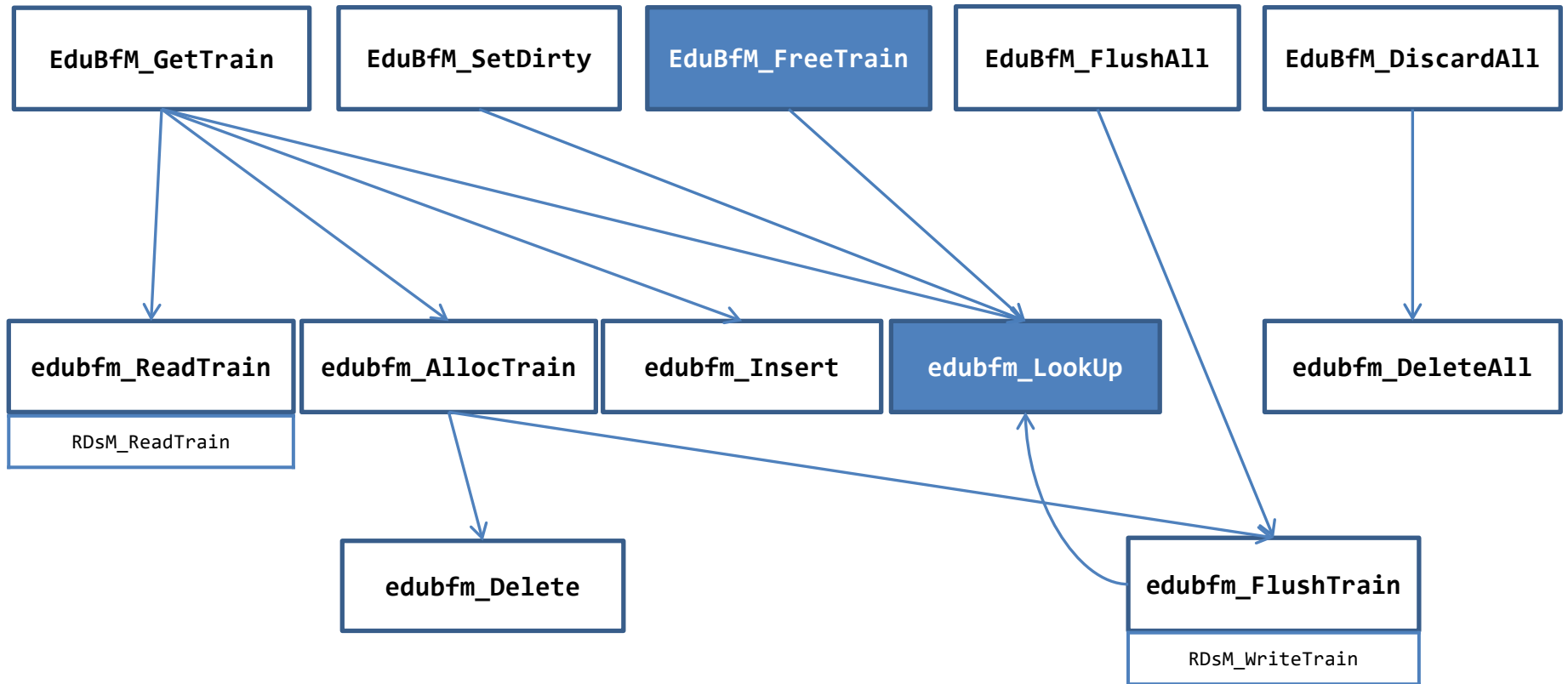
EduBfM_GetTrain



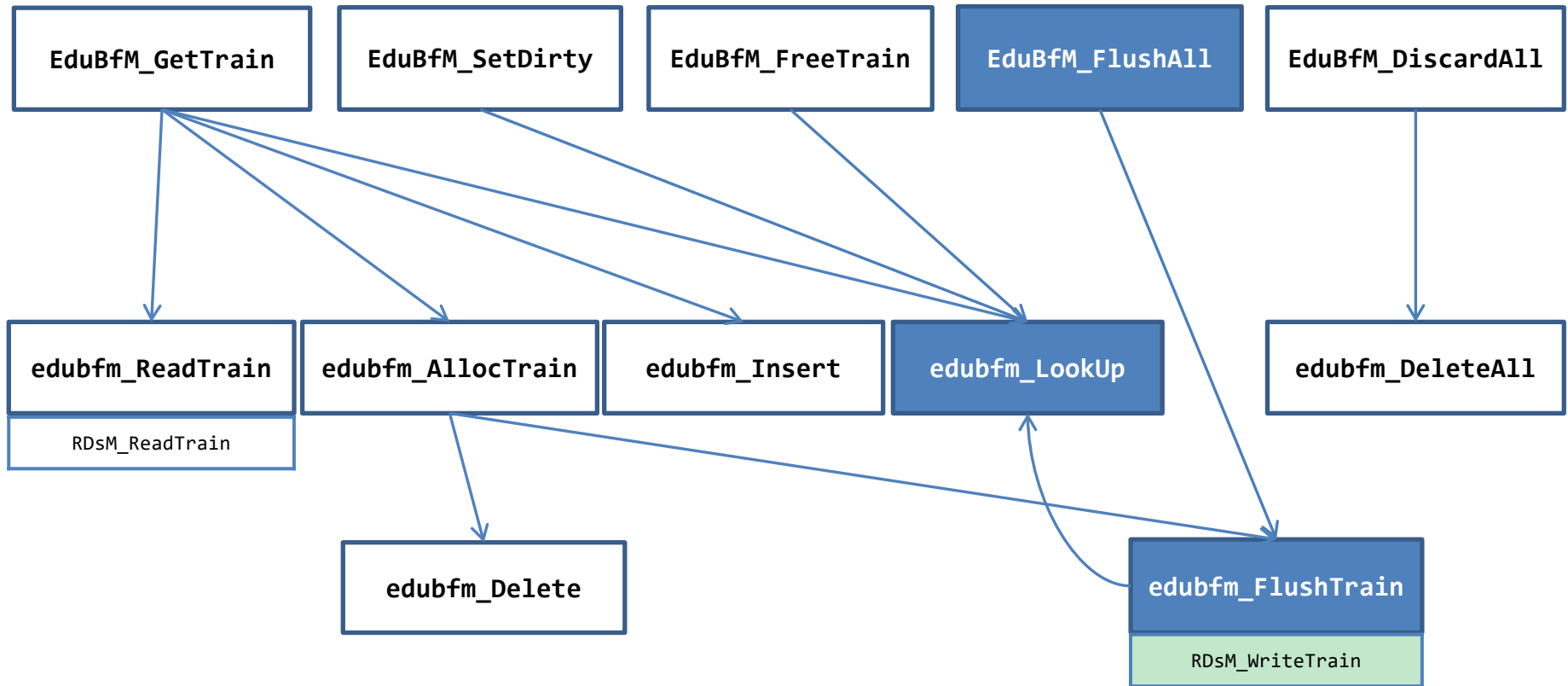
EduBfM_SetDirty



EduBfM_FreeTrain



EduBfM_FlushAll



EduBfM_DiscardAll

