Project2 report

20160463 성해빈

구현한 함수

EduBfM_GetTrain

```
Four EduBfM_GetTrain(
    TrainID             *trainId,               /* IN train to be used */
    char                **retBuf,               /* OUT pointer to the returned
buffer */
    Four                type )                  /* IN buffer type */
{
    Four                e;                      /* for error */
    Four                index;                  /* index of the buffer pool */


    /*@ Check the validity of given parameters */
    /* Some restrictions may be added         */
    if(retBuf == NULL) ERR(eBADBUFFER_BFM);

    /* Is the buffer type valid? */
    if(IS_BAD_BUFFERTYPE(type)) ERR(eBADBUFFERTYPE_BFM);


    index = edubfm_LookUp(trainId, type);
    if(index == NOTFOUND_IN_HTABLE){ //the train does not exist in the pool
        index = edubfm_AllocTrain(type);
        if(index < 0) ERR(index);

        e = edubfm_ReadTrain(trainId, BI_BUFFER(type, index), type);
        if(e < 0) ERR(e);

        BI_KEY(type, index).pageNo = trainId->pageNo;
        BI_KEY(type, index).volNo = trainId->volNo;
        BI_FIXED(type, index) = 1;
        BI_BITS(type, index) |= REFER;

        e = edubfm_Insert(&BI_KEY(type, index), index, type);
        if(e < 0) ERR(e);

    }
    else{  //train in hashtable === train already exist in the pool
        BI_FIXED(type, index) += 1;
        BI_BITS(type, index) |= REFER;
    }

    *retBuf = BI_BUFFER(type, index);

    return(eNOERROR);   /* No error */

}  /* EduBfM_GetTrain() */
```

해시 테이블에 페이지가 있다면 받아오고, 아니라면 새로 할당을 해서 해시 테이블에 넣는다.

EduBfM_FreeTrain

```
Four EduBfM_FreeTrain(
    TrainID             *trainId,       /* IN train to be freed */
    Four                type)           /* IN buffer type */
{
    Four                index;          /* index on buffer holding the train */
    Four        e;      /* error code */

    /*@ check if the parameter is valid. */
    if (IS_BAD_BUFFERTYPE(type)) ERR(eBADBUFFERTYPE_BFM);

    index = edubfm_LookUp(trainId, type);
    if(index == NOTFOUND_IN_HTABLE){
        ERR(eNOTFOUND_BFM);
    }
    else{
        BI_FIXED(type, index) -= 1;

        if(BI_FIXED(type, index) < 0){
            printf("fixed counter is less than 0!!!\n");
            printf("trainId = {%d,  %d}\n", trainId->volNo, trainId->pageNo);
            BI_FIXED(type, index) = 0;
        }

    }

    return( eNOERROR );

} /* EduBfM_FreeTrain() */
```

fixed를 내리기만 한다.
그 이유는 eviction을 할 때 disk write을 하면 되기 때문이다.
fixed가 0보다 작은 값으로 내려갈 때의 경고는 solution output이랑 맞췄다.

EduBfM_SetDirty

```
Four EduBfM_SetDirty(
    TrainID             *trainId,               /* IN which train has been
modified in the buffer?  */
    Four                type )                  /* IN buffer type */
{
    Four                index;                  /* an index of the buffer table
& pool */


    /*@ Is the paramter valid? */
    if (IS_BAD_BUFFERTYPE(type)) ERR(eBADBUFFERTYPE_BFM);

    index = edubfm_LookUp(trainId, type);
    if(index == NOTFOUND_IN_HTABLE){
        ERR(eNOTFOUND_BFM);
    }
    else{
```

```
        BI_BITS(type, index) |= DIRTY;
    }

    return( eNOERROR );

}  /* EduBfM_SetDirty */
```

buffer page가 disk랑 내용이 다르다는 것을 마킹해놓는다.

EduBfM_FlushAll

```
Four EduBfM_FlushAll(void)
{
    Four        e;                      /* error */
    Two         i;                      /* index */
    Four        type;                   /* buffer type */

    for(type = 0; type < NUM_BUF_TYPES; type++){
        for(i = 0; i < BI_NBUFS(type); i++){
            if(BI_BITS(type, i) & DIRTY){
                e = edubfm_FlushTrain(&BI_KEY(type, i), type);
                if(e < 0) ERR(e);
            }
        }
    }

    return( eNOERROR );

}  /* EduBfM_FlushAll() */
```

buffer을 전부 디스크와 동기화한다.

EduBfM_DiscardAll

```
Four EduBfM_DiscardAll(void)
{
    Four    e;          /* error */
    Two     i;          /* index */
    Four    type;           /* buffer type */

    for(type = 0; type < NUM_BUF_TYPES; type++){
        //bufTable
        for(i = 0; i < BI_NBUFS(type); i++){
            BI_KEY(type, i).pageNo = NIL;
            BI_BITS(type, i) = 0;
        }
    }

    //hashTable
    e = edubfm_DeleteAll();
    if(e < 0) ERR(e);

    return(eNOERROR);

}  /* EduBfM_DiscardAll() */
```

buffer table과 hash table을 삭제한다.

edubfm_ReadTrain

```
Four edubfm_ReadTrain(
    TrainID *trainId,       /* IN which train? */
    char    *aTrain,        /* OUT a pointer to buffer */
    Four    type )      /* IN buffer type */
{
    Four e;         /* for error */


    /* Error check whether using not supported functionality by EduBfM */
    if (RM_IS_ROLLBACK_REQUIRED()) ERR(eNOTSUPPORTED_EDUBFM);

    e = RDsM_ReadTrain(trainId, aTrain, BI_BUFSIZE(type));
    if(e < 0) ERR(e);

    return( eNOERROR );

}  /* edubfm_ReadTrain */
```

page를 disk에서부터 읽고 buffer로 쓴다.
이 부분은 RDsM 모듈의 역할이기 때문에 사실상 delegation 말고는 할 게 없다.

edubfm_AllocTrain

```
Four edubfm_AllocTrain(
    Four    type)           /* IN type of buffer (PAGE or TRAIN) */
{
    Four    e;          /* for error */
    Four    victim;         /* return value */
    Four    i;


    /* Error check whether using not supported functionality by EduBfM */
    if(sm_cfgParams.useBulkFlush) ERR(eNOTSUPPORTED_EDUBFM);

    /* choose target buffer element by second chance buffer replacement algorithm
*/
    victim = BI_NEXTVICTIM(type);

    if(BI_NBUFS(type) <= 0){
        ERR(eNOUNFIXEDBUF_BFM);
    }

    i = 0;
    while(1){
        if (BI_FIXED(type, victim) == 0) {
            if(BI_BITS(type, victim) & REFER){ //if REFER bit is 1
                BI_BITS(type, victim) ^= REFER; //set REFER bit to 0
            }
            else{ //if REFER bit is 0
                break;
            }
        }
```

```
        i++;
        victim = (victim + 1) % BI_NBUFS(type);
        //nbufs : total count of buffer element
        //if reached to end, go to the start by %

    }


    /* initialize chosen target buffer element */


    if(BI_KEY(type, victim).pageNo != NIL){ //the buffer element is in the
hashtable (linked list)
        if(BI_BITS(type, victim) & DIRTY){
            e = edubfm_FlushTrain(&BI_KEY(type, victim), type);
            if(e < 0)
                ERR(e);
        }

        e = edubfm_Delete(&BI_KEY(type, victim), type);
        if(e < 0)
            ERR(e);
    }

    BI_BITS(type, victim) = 0;
    BI_NEXTVICTIM(type) = (victim + 1) % BI_NBUFS(type);


    return( victim );

}  /* edubfm_AllocTrain */
```

마치 cache eviction 을 짜듯이, buffer에 넣을 page를 위한 공간을 찾아낸다.
fixed가 1 이상이면 이미 누가 쓰고 있는 것이므로,
second chance replacement를 써서 REFER bit가 0인 애를 축출해낸다. 1인 애는 한 번 살려주는 의미에서 0으로 세팅한다.
또 축출당한 친구(victim)이 할당이 돼있었다면 할당 해제를 해준다.

edubfm_Insert

```
Four edubfm_Insert(
    BfMHashKey      *key,           /* IN a hash key in Buffer Manager */
    Two         index,          /* IN an index used in the buffer pool */
    Four        type)           /* IN buffer type */
{
    Two         hashValue;


    CHECKKEY(key);    /*@ check validity of key */

    if( (index < 0) || (index > BI_NBUFS(type)) )
        ERR( eBADBUFINDEX_BFM );

    hashValue = BFM_HASH(key, type);
```

```
    //always insert in head of linked list
    BI_NEXTHASHENTRY(type, index) = BI_HASHTABLEENTRY(type, hashValue);
    //HashTableEntry would be -1 if no collision, which is perfectly fine
    BI_HASHTABLEENTRY(type, hashValue) = index;


    return( eNOERROR );

}  /* edubfm_Insert */
```

hash table insert다.
hash table entry에 있는 값을 현재 넣을 buffer Table에다 next hash entry로 넣고, 현재 넣을 buffer를 새로운 hash table entry로 대체한다.
마치 linked list의 head만 hash table entry로 관리한다 생각하면 된다.
처음에는 hash table entry가 -1이다.

edubfm_Delete

```
Four edubfm_Delete(
    BfMHashKey          *key,                   /* IN a hash key in buffer
manager */
    Four                type )                  /* IN buffer type */
{
    Two                 i, prev;
    Two                 hashValue;


    CHECKKEY(key);    /*@ check validity of key */

    hashValue = BFM_HASH(key, type);
    prev = NIL;
    i = BI_HASHTABLEENTRY(type, hashValue);

    while(1) {

        if(i == NIL)
            ERR( eNOTFOUND_BFM );

        if(EQUALKEY(&BI_KEY(type, i), key))
            break;

        prev = i;
        i = BI_NEXTHASHENTRY(type, i);
    }

    if(prev == NIL) //first entry match
        BI_HASHTABLEENTRY(type, hashValue) = BI_NEXTHASHENTRY(type, i);
    else
        BI_NEXTHASHENTRY(type, prev) = BI_NEXTHASHENTRY(type, i);

    return( eNOERROR );

}  /* edubfm_Delete */
```

hash delete다.
linked list delete를 할 때처럼 link에만 조심해주면 된다.

edubfm_Deleteall

```
Four edubfm_DeleteAll(void)
{
    Two     type, i;

    for(type = 0; type < NUM_BUF_TYPES; type++){
        for(i = 0; i < HASHTABLESIZE(type); i++){
            BI_HASHTABLEENTRY(type, i) = NIL;
        }
    }


    return(eNOERROR);

} /* edubfm_DeleteAll() */
```

hash table의 entry, 즉 linked list의 head들만 전부 NIL로 만들어주면 된다.

edubfm_LookUp

```
Four edubfm_LookUp(
    BfMHashKey          *key,                   /* IN a hash key in Buffer
Manager */
    Four                type)                   /* IN buffer type */
{
    Two                 i;                  /* indices */
    Two                 hashValue;


    CHECKKEY(key);    /*@ check validity of key */

    hashValue = BFM_HASH(key, type);
    i = BI_HASHTABLEENTRY(type, hashValue);

    while(1){
        if(i == NIL)
            return(NOTFOUND_IN_HTABLE);

        if(EQUALKEY(&BI_KEY(type, i), key))
            break;

        i = BI_NEXTHASHENTRY(type, i);
    }

    return i;

}  /* edubfm_LookUp */
```

hash lookup이다.
그냥 linked list sequential search O(n)이다.

edubfm_FlushTrain

```
Four edubfm_FlushTrain(
    TrainID             *trainId,       /* IN train to be flushed */
    Four            type)           /* IN buffer type */
{
    Four            e;          /* for errors */
    Four            index;          /* for an index */


    /* Error check whether using not supported functionality by EduBfM */
    if (RM_IS_ROLLBACK_REQUIRED()) ERR(eNOTSUPPORTED_EDUBFM);

    index = edubfm_LookUp(trainId, type);
    if(index == NOTFOUND_IN_HTABLE){
        ERR(eNOTFOUND_BFM);
    }
    else{
        if(BI_BITS(type, index) & DIRTY){
            e = RDsM_WriteTrain(BI_BUFFER(type, index), trainId,
BI_BUFSIZE(type));
            if(e < 0) ERR(e);

            BI_BITS(type, index) ^= DIRTY; //reset dirty bit
        }
    }


    return( eNOERROR );

}  /* edubfm_FlushTrain */
```

dirty bit가 세팅되어 있으면 disk write를 해준다.

느낀 점 discussion

Github에 있는 KAIST쪽의 업데이트된 매뉴얼을 보았더니 설명이 좀 더 상세히 나와있어서 좋았다.
특히 제공되는 function 사용법에 대한 example code가 있어서 좋았다.
https://github.com/odysseus-educosmos/ODYSSEUS-EduCOSMOS

마치 cache를 구현하는 것 같다. 그냥 언어도 그렇고, 형식도 그렇고 좀 핀토스같다.
OS쪽에서도 마찬가지로 file system을 buffer cache에서 관리하는 것으로 알고 있는데,
OS에서 매니지하는 buffer cache와 BfM이 결합해서 어떤 기작이 될 지 궁금하다.

전체적으로 설명이 너무 부실한 것 같다. 혼자 뻘짓으로 알아낸 게 많다.
그나마 솔루션 output이 제공되어서 어떻게 코딩해야하는지에 대한 참고가 되었다.

checker.sh로 내 output과 solution output을 diff하게 만들었다.
이런 걸 만들 필요 없이 알아서 자동으로 채점해주면 좋긴 하겠다.