박준호 김동민 윤지아

PHYSIO-LOGICAL LOGGING

CONTENTS

- no-force with checkpoint
- physiological logging and recovery
- insert example
- overflow exception
- conclusion

NO-FORCE WITH CHECK POINT

ORIGINAL SQLITE

- sqlite3BtreeCommitPhaseOne과
  sqlite3BtreeCommitPhaseTwo를 차례대로 호출함

- commitPhaseTwo에서 하나의 transition에 대해 disk에 쓰는 과정을 진행함

- 우리는 checkpoint를 두고 일정 수준 이상의 log가 쌓이면 이들을 한꺼번에 disk에 쓰도록 구현(log force at commit)

IMPLEMENTATION

src/btreeInt.h

```
void sqlite3Log(Pgno pgno,int opcode, int redo_size, const char *redo_log, int undo_size,const
char *undo_log){
……
        p_check++;
……
}
```

src/btree.c

```
int sqlite3BtreeCommitPhaseOne(Btree *p, const char *zMaster){
  int rc = SQLITE_OK;
  if(p_check >= 1000 || pragma_check >= 1){ //pragma 명령어일 경우 예외 처리
……
  }
  return rc;
}
```

src/btree.c

```
int sqlite3BtreeCommitPhaseTwo(Btree *p, int bCleanup){
  if(p_check >= 1000 ||  pragma_check >=1){ //pragma 명령어일 경우 예외 처리
……
    p_check = 0;
……
  }
}
```

PHYSIOLOGICAL LOGGING AND RECOVERY

PYSIOLOGOCAL LOGGING

- 쿼리 결과를 물리적으로 디스크에 저장하지 않고 로그파일에 저장
  - database가 open될 때 [database].log 파일을 생성 및 open
  - mmap 함수를 이용한 memory mapping
  - msync를 이용하여 file에 쓰기 가능

PYSIOLOGOCAL LOGGING - LOG FILE 생성 및 MAPPING

src/btreeInt.h

```
int sqlite3_open(const char *zFilename, sqlite3 **ppDb){
        char logFilename[40];
        strncpy(logFilename,zFilename,36);
        int logNameLen = strlen(logFilename);
        strcat(logFilename, ".log");
        log_fd = open(logFilename, O_RDWR | O_CREAT, 0644);
        if(log_fd < 0){
                fprintf(stderr, "LOG FILE OPEN ERROR\n");
        }else{
                ftruncate(log_fd,1024*4096);
        }
        origin_log_buffer = log_buffer = (void*) mmap(NULL, 1024*4096, PROT_READ | PROT_WRITE,
MAP_SHARED, log_fd,0);
        if(log_buffer == MAP_FAILED){
                fprintf(stderr, "LOG FILE MAPPING ERROR\n");
        }
        return openDatabase(zFilename, ppDb,
                        SQLITE_OPEN_READWRITE | SQLITE_OPEN_CREATE, 0);
}
```

PYSIOLOGOCAL LOGGING

- query가 호출되면(예를 들어 insert) redo_log와 undo_log를 남김
  - newcell(or oldcell)의 내용과 그 크기, 해당 cell의 pageno, idx, opcode 등의 parameter을 이용하여 sqliteLog 함수 호출
- sqliteLog 함수는 이들을 구조에 맞게 memory에 작성
- 만약 commit이 발생하면 file로 msync

PYSIOLOGOCAL LOGGING - WRITE LOG(INSERT QUERY일 경우)

src/btree.c

```
int sqlite3BtreeInsert(BtCursor *pCur, const BtreePayload *pX, int appendBias, int seekResult)
{
      ……
  redo_s = szNew + sizeof(int);
  redo_log = (char*)malloc(redo_s);
  memcpy(redo_log, &idx, sizeof(int));
  memcpy(redo_log+sizeof(int), newCell, redo_s-sizeof(int));

  if(loc != 0){
          undo_s = 0;
          undo_log = (char*)malloc(1);
  }

  sqlite3Log(pPage->pgno, loc==0?2:1, redo_s, redo_log, undo_s, undo_log);
  free(redo_log);
  free(undo_log);

  insertCell(pPage, idx, newCell, szNew, 0, 0, &rc);
     ……
```

PYSIOLOGOCAL LOGGING – WRITE LOG(INSERT QUERY일 경우)

src/btreeInt.h

```
void sqlite3Log(Pgno pgno,int opcode, int redo_size, const char *redo_log, int undo_size,const char *undo_log){
……
        int log_size = sizeof(Pgno) + sizeof(int)*3 + sizeof(int)*2 + undo_size + redo_size;
        void* log = malloc(log_size);
        p_check++;
        int tmp_size = 0;
        memcpy(log+tmp_size, &log_size, sizeof(int));
        tmp_size+=sizeof(int);
        memcpy(log+tmp_size, &lastLsn, sizeof(int));
        tmp_size+= sizeof(int);
        lastLsn+=log_size;
        memcpy(log+tmp_size, &opcode, sizeof(int));
        tmp_size+= sizeof(int);
        memcpy(log+tmp_size, &pgno, sizeof(Pgno));
        tmp_size+= sizeof(Pgno);
        memcpy(log+tmp_size, &redo_size,sizeof(int));
       tmp_size+= sizeof(int);
        memcpy(log+tmp_size, redo_log, redo_size);
        tmp_size+= redo_size;
        memcpy(log+tmp_size, &undo_size,sizeof(int));
        tmp_size+= sizeof(int);
        memcpy(log+tmp_size, undo_log, undo_size);
        memcpy(log_buffer, log, log_size);
        if(opcode == 4){
                msync(old_log_buffer, log_buffer - old_log_buffer + log_size , MS_SYNC);
                old_log_buffer = log_buffer + log_size;
        }
……
};
```

log structure							
log_size	lastLsn	opcode	pgno	redo_size	redo_log	undo_size	undo_log

RECOVERY

- checkpoint가 일정 수준에 도달하지 못하고 어떤 이유로든 종료되는 경우 database의 정보의 복구를 위해 recovery가 필요함
- 데이터베이스의 정보가 초기화되는 opendatabase에서 log파일을 parsing하여 복구를 진행
  - 반복문을 통해 log정보를 parsing
  - 초기화된 database로 부터 newCell과 page를 얻은 후
  - log에 남겨진 정보를 바탕으로 insertCell을 실행
  - 복구된 정보들이 commit이 되도록 dirty page를 만들어줌

RECOVERY(INSERT QUERY일 경우)

src/main.c

```
static int openDatabase(const char *zFilename, sqlite3 **ppDb, unsigned int flags, const char *zVfs){
……
  is_open = 1;                                    // is_open이 1일 경우 이후 실행되는 쿼리에 대해 log를 남기지 않음
  char tempsql0[100]="PRAGMA journal_mode=wal;";
  char tempsql1[100]="select * from test;";       // btree의 초기화를 위해 한번 이상의 쿼리가 필요함
  sqlite3_exec(db,tempsql0,0,0, &zErrMsg);
  sqlite3_exec(db,tempsql1,0,0, &zErrMsg);
  is_open = 0;

  int lastLsn, log_size, opcode, redo_size, undo_size, idx;
  char * redo_log, *undo_log;
  MemPage* pPage;
  u8* newCell;
  Pgno pgno;
  while(1){
         memcpy(&log_size,log_buffer,sizeof(int));
        if(log_size == 0)                          //log_size가 0일 경우 파일이 끝난 것으로 판단
                break;
         log_buffer+=sizeof(int);
         memcpy(&lastLsn,log_buffer,sizeof(int));
         log_buffer+=sizeof(int);
         memcpy(&opcode,log_buffer,sizeof(int));
         log_buffer+=sizeof(int);
         memcpy(&pgno,log_buffer, sizeof(Pgno));
         log_buffer+=sizeof(Pgno);
         memcpy(&redo_size,log_buffer,sizeof(int));
         log_buffer+=sizeof(int);
         redo_log = (char*)malloc(sizeof(char)*redo_size);
         memcpy(redo_log, log_buffer,sizeof(char)*redo_size);
         log_buffer+=sizeof(char)*redo_size;
         memcpy(&undo_size,log_buffer,sizeof(int));
         log_buffer+=sizeof(int);
         undo_log = (char*)malloc(sizeof(char)*undo_size);
         memcpy(undo_log, log_buffer,sizeof(char)*undo_size);
         log_buffer+=sizeof(char)*undo_size;
         if(opcode != 1)
                continue;                           // insert 문에 관해서만 처리
…(뒷장 연속)
```

RECOVERY(INSERT QUERY일 경우)

src/main.c

```
……
            sqlite3BtreeEnter(db->aDb[0].pBt);                                  //btree enter
            allocateTempSpace(db->aDb[0].pBt->pBt);
            newCell = db->aDb[0].pBt->pBt->pTmpSpace;                           //newCell 할당
            btreeGetPage(db->aDb[0].pBt->pBt, pgno, &(pPage), 0);               //pgno로 부터 page를 얻어옴


            memcpy(&idx, redo_log, sizeof(int));
            memcpy(newCell, redo_log + sizeof(int), redo_size - sizeof(int));   // redo_log parsing

            insertCell(pPage, idx, newCell, redo_size-sizeof(int), 0, 0, &rc);  // log로 부터 parsing한 정보를 바탕으로 page에 insert
            sqlite3BtreeLeave(db->aDb[0].pBt);                                  //btree leave
            pPage->pDbPage->pPager->eState = PAGER_WRITER_FINISHED;
            sqlite3PcacheMakeDirty(pPage->pDbPage);                             // disk에 쓰기 위해 dirty page로 만듬
    }

    db->aDb[0].pBt->inTrans=TRANS_WRITE;                                        //disk write flag 설정
    pragma_check = 1;                                                           //pragma_check를 이용하여 p_check와는 무관하게 commit
    is_open = 1;
    sqlite3_exec(db,tempsql0,0,0, &zErrMsg);                                    //pragma 명령어를 실행하여 강제로 commit
    is_open = 0;
    db->aDb[0].pBt->inTrans=TRANS_NONE;                                         // disk write flag 해제
    log_buffer = origin_log_buffer;
    memset(log_buffer, 0x00, 1024*4096);                                        // log 파일 초기화
    msync(log_buffer, 1024*4096, MS_SYNC);
……
}
```

INSERT EXAMPLE

TABLE 설정 및 데이터 입력

```
ga@ubuntu:~/sqlite/sqlite3_group2/sqlite_new_version$ ./sqlite3 test.db
REcovery log 24
REcovery log 0
SQLite version 3.14.1 2016-08-11 18:53:32
Enter ".help" for usage hints.
sqlite> create table test (a int, b int);
log : 0 1 1 12 0 36 0
log : 36 1 2 58 0 82 1
log : 118 0 4 0 0 24 2
sqlite> pragma journal_mode = wal;          // pragma 명령어를 사용하여 임의로 sync 발생
wal
log : 142 0 4 0 0 24 0
sqlite> insert into test values ( 3, 4);
log : 166 2 1 11 0 35 1
log : 201 0 4 0 0 24 2
sqlite> insert into test values ( 4, 5);
log : 225 2 1 11 0 35 3
log : 260 0 4 0 0 24 4
sqlite> select * from test;
3|4
4|5
log : 284 0 4 0 0 24 5                      //p_check(마지막 인자)가 1000보다 작기 때문에 log file에는
                                            //쓰여졌지만 disk에 sync되진 않았음
```

NO-FORCE 및 복구 확인

```
ga@ubuntu:~/sqlite/sqlite3_group2/sqlite_new_version$ mv test.db.log test_.db.log
//parsing 할 수 있는 지정된 log파일을 다른 곳으로 옮김(recovery가 안되도록 설정)
ga@ubuntu:~/sqlite/sqlite3_group2/sqlite_new_version$ ./sqlite3 test.db
REcovery log 0
SQLite version 3.14.1 2016-08-11 18:53:32
Enter ".help" for usage hints.
sqlite> select * from test;              // 저장된 데이터 없음 및 no-force 확인
log : 0 0 4 0 0 24 0
sqlite> ^C^C^C
ga@ubuntu:~/sqlite/sqlite3_group2/sqlite_new_version$ mv test_.db.log test.db.log
//insert에 대한 정보가 닮긴 log 파일을 정상적인 위치에 복구
ga@ubuntu:~/sqlite/sqlite3_group2/sqlite_new_version$ ./sqlite3 test.db
REcovery log 24
REcovery log 35
REcovery log 24
REcovery log 35
REcovery log 24
REcovery log 24
REcovery log 0
SQLite version 3.14.1 2016-08-11 18:53:32
Enter ".help" for usage hints.
sqlite> select * from test;              //복구 확인
3|4
4|5
log : 0 0 4 0 0 24 0
sqlite>
```

OVERFLOW EXCEPTION

OVERFLOW AND BALANCING

- 페이지가 가득차는 경우에 split이 발생하여 한 테이블에서 쓰는 페이지가 늘어나게 됨.
- 이런 경우에 pgno가 달라지기 때문에 physiological logging이 어려워지는 문제가 발생함
- 따라서 우리는 split이 발생하는 쿼리문에서 바로 checkpoint를 시행하여 문제를 생략함

OVERFLOW EXCEPTION

src/main.c

```
3141.int sqlite3BtreeInsert(BtCursor *pCur, const BtreePayload *pX, int appendBias, int seekResult){
3142.     ……
8120.  pCur->info.nSize = 0;
8121.  if( pPage->nOverflow ){
8122.    assert( rc==SQLITE_OK );
8123.    pCur->curFlags &= ~(BTCF_ValidNKey);
8124.    rc = balance(pCur);
8125.
8126.    printf("\n***overflow***\n");
8127.    /* Must make sure nOverflow is reset to zero even if the balance()
8128.    ** fails. Internal data structure corruption will result otherwise.
8129.    ** Also, set the cursor state to invalid. This stops saveCursorPosition()
8130.    ** from trying to save the current position of the cursor.  */
8131.    pCur->apPage[pCur->iPage]->nOverflow = 0;
8132.    pCur->eState = CURSOR_INVALID;
8133.    pragma_check = 3;
8134.  }
8135.  assert( pCur->apPage[pCur->iPage]->nOverflow==0 );
8136.     ……
```

CONCLUSION

CONCLUSION

- 기존에 newcell로 부터 pasing한 정보를 가지고 있는 cellinfo와 그 크기를 이용하여 recovery 과정에서 fillInCell을 하기 위해 다양한 변수들을 건드리다보니 이 값들이 일부 바뀌는 현상이 발견되었다.
- 결과적으로 다른 팀들과 같은 방법을 이용하여 구현하였으나 cell에 대한 구체적인 동작 방법을 분석하고 좀 더 많이 이해를 할 수 있는 과정이었다.