데이터베이스 시스템 project - final

20174744 이 동 현

[언어/환경]

구현 언어 = C(C++)

구현 ide : CLION

운영체제 : Windows 10

*또한 시연 비디오를 담아 두었습니다 !

[구체적인 구현 내용]

지난 보고서(0506)에서 소개한 것과 거의 동등하게 실제로 구현 하였습니다.

우선 기본적인 구현 setting은 **'수업 때 다룬 것 그대로'** 하였습니다.

(0) 메인 메뉴 출력

1을 누르면 테이블 생성

2를 누르면 레코드 삽입

3을 누르면 레코드 검색과 (primary key)

그 이후에 원하면 그 record 안에서 특정 column을 질의할 수 있습니다.

```
====================================
Insert Number!
1. Create Table
2. Insert record
3. Search record & Search Column
====================================
```

또한 loop문으로 계속 메뉴를 띄우는게 아니라, 한 번에 하나의 연산만 하고 프로그램이 종료되는 방식으로 구현하였습니다. 이렇게 하면 input file(txt 파일)을 읽는다는 것이 확실히 증명이 되기 때문입니다.

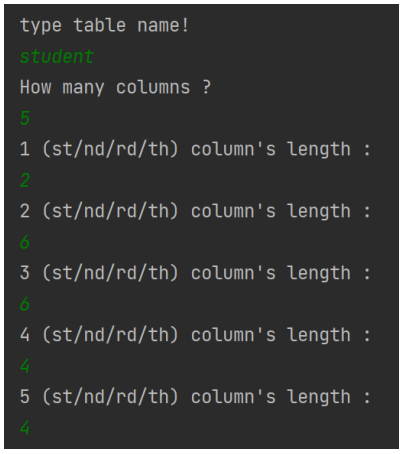
(1) 테이블 생성

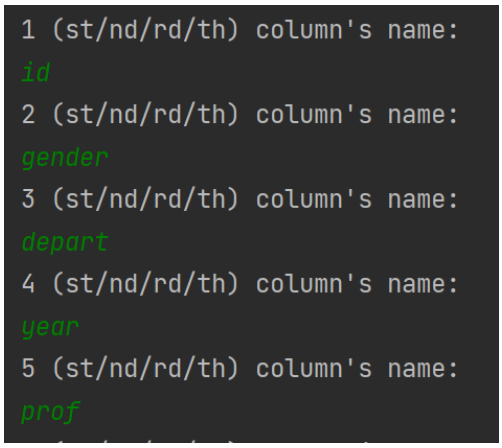
***테이블 저장 장소(txt 파일) : /DBS/cmake-build-debug/tables/ 에 저장 되어 있습니다(또는 , 새로 생성됩니다.).**

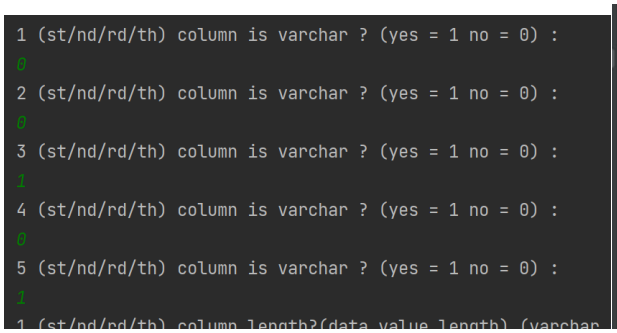
다음과 같은 예시 테이블(student)로 실행하며 설명을 드릴 생각입니다.

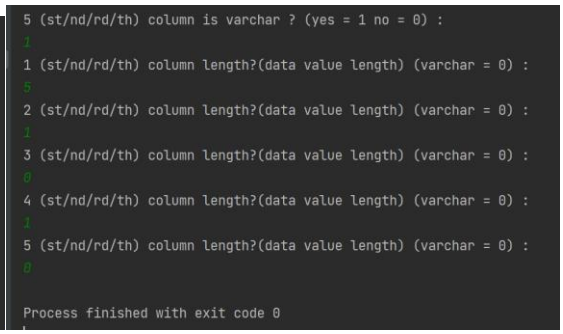
컬럼 이름	크기
ID	5byte
gender	1byte
depart	가변길이
year	1byte
prof	가변길이

Student 테이블을 만들면 student.txt 라는 이름으로 텍스트 파일이 생성됩니다.



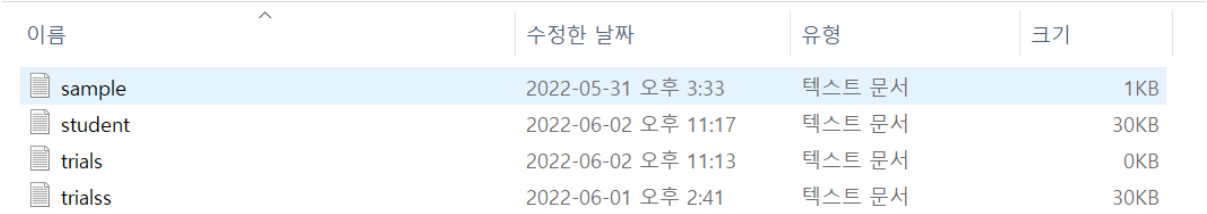






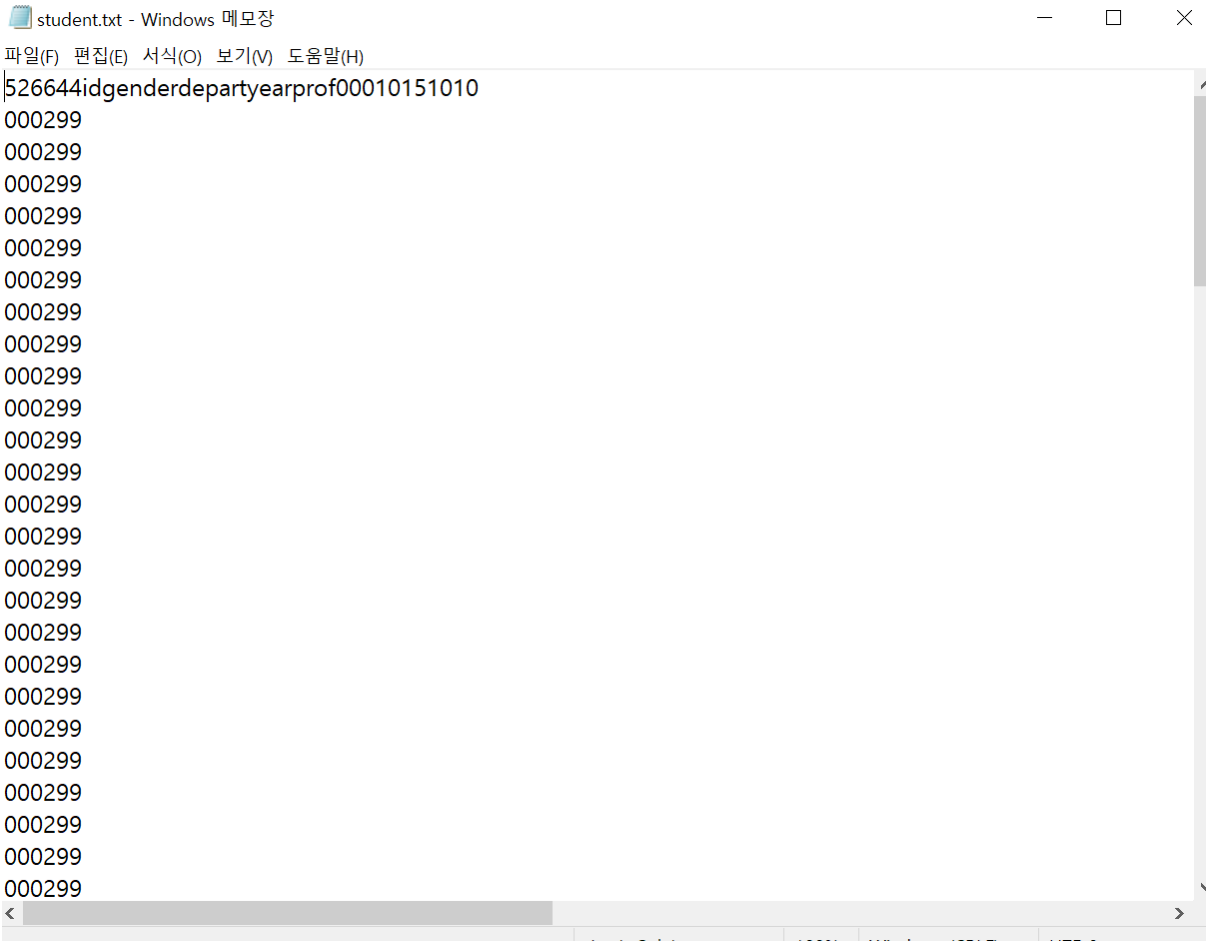
다음과 같이 입출력을 받습니다(메타데이터 생성)

차례대로 column이름길이(이름 자체의 길이 – 예 id 면 2) , column 이름 , varchar여부, 고정길이 column이면 길이는 몇인지 를 받습니다.



이름	수정한 날짜	유형	크기
sample	2022-05-31 오후 3:33	텍스트 문서	1KB
student	2022-06-02 오후 11:17	텍스트 문서	30KB
trials	2022-06-02 오후 11:13	텍스트 문서	0KB
trialss	2022-06-01 오후 2:41	텍스트 문서	30KB

그럼 이렇게 student.txt 파일이 생깁니다.



안을 열어보면 다음과 같이 아까 입력했던 값을 기반으로 metadata 가 형성되고

아랫줄에 empty block 들이 생성됩니다. **(block 크기 300byte + 개행 2byte)**

여기서 (000299 의미 -> record수 0 개 / free space pointer = 299)

*참고 : 예외처리 (이미 있는 테이블인데 또 만들려고 하는 경우) 하였습니다.

```
==================================
1
type table name!
trialss
table name already exists.... try something else!

Process finished with exit code 0
```

(코드)

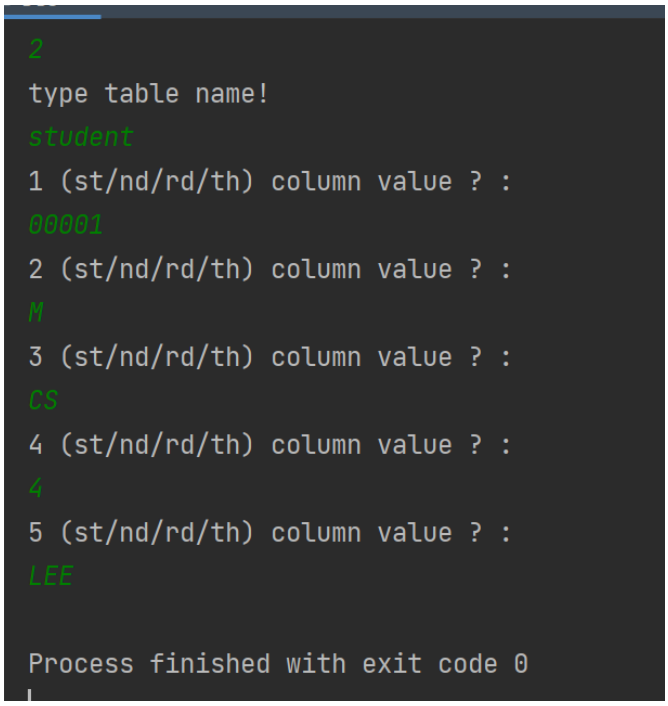
```
FILE * fp;
if(now_menu == 1) {
    //create table
    printf( format: "type table name!\n");
    std::string now_finding;
    std::cin >> now_finding;
    std::string temp_directory = ".\\tables\\" + now_finding + ".txt";

    if (fp = fopen( Filename: temp_directory.c_str(), Mode: "r")) {
        //이미 테이블 이름이 있음 ( 못만든다)
        printf( format: "table name already exists.... try something else!\n");
    } else {
        //새로운 테이블 이름
        fp = fopen( Filename: temp_directory.c_str(), Mode: "w");
        std::string now_meta = insert_metadata();
        fwrite( Str: now_meta.c_str(), Size: now_meta.length(), Count: 1, File: fp);

        std::string blank = "000299";
        blank = make_one_block( sample: blank);
        for(int i=0; i<100; i++) {
            fwrite( Str: blank.c_str(), Size: blank.length(), Count: 1, File: fp);
        }
    }
}else if(now_menu == 2){
```

(2) 레코드 삽입

2번 메뉴를 누르면 , 일단 메타데이터를 읽습니다.

그래서 데이터가 몇 개 있는지 파악한 후, 개수에 맞추어 column값으로 어떤 것을 넣을지 입력을 받습니다. 이 테이블 말고도 , **첫 KEY 를 pk로 하고 자 하며 이름은 id, (5글자) 로 하고자 합니다.**

B+ 트리를 공부할 때에 DBS 자체에 PK(id) 값을 지정하고(막상 테이블에는 pk 가 없음에도 불구하고,) 데이터를 관리하는 경우도 많다는 것을 들었 어서, 이러한 구현 방식에는 문제가 없다고 생각합니다.



(입력 받는 모습)

그러면 (*코드 참고) 수업 때 배운 것과 매우 비슷하게,.

페이지(블록) 을 한 개 씩 읽어서 일단 free space가 충분히 있는 지 계산합니다.

Free space가 부족하면, 다음 블록을 읽습니다(loop 로 찾을 때 까지 반복.)

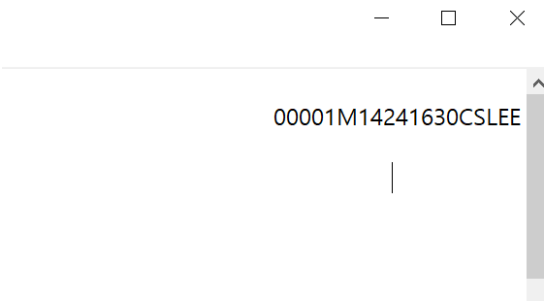
Free space가 충분한 page를 발견하면, 그곳에 왼쪽에는 record 크기 / 시작점(pointer)

를 각 3바이트 (총 6바이트) 로 저장합니다. 우측에는 variable length records 를 저장합니다.

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

526644idgenderdepartyearprof00010151010
001280019280
000299
000299
000299
000299
000299
000299
000299

왼쪽에는 순서대로 record갯수 / end_of_free_space offset / (size of that block / size of that block) (모두 크기는 3글자 – 3byte) 가 구성이 되어있는 것을 볼 수 있고



우측에는 아까 입력했던 값을 기반으로 variable record가 잘 들어갔음을 알 수 있습니다.

다만, column value 크기는 10이 넘으면 안되겠습니다. 1byte로 (0~9) 로 받아서 처리하기 때문 입니다.

*free space 계산

End of free space – ((record 수+1) * 6) >= (이번에 넣을 record 의 크기 + 6)

```
bool avail_free_space(std::string info , int len){
    std::string sub_str = info.substr( pos: 3, n: 3);
    int end_free_space = std::stoi( str: sub_str);

    std::string sub_str2 = info.substr( pos: 0, n: 3);
    int start_free_space = (std::stoi( str: sub_str2)+1) * 6;
    //printf("%d %d\n", end_free_space, start_free_space);

    int remain = end_free_space - start_free_space;

    if(remain > len + 6) return true;
    else return false;
}
```

* 레코드 삽입 코드

```
    }
}else if(now_menu == 2){
    //레코드 삽입
    std::string now_finding = "trialss";
    //std::cin >> now_finding;
    std::string temp_directory = ".\\tables\\" + now_finding +".txt";
    fp = fopen( Filename: temp_directory.c_str(), Mode: "r+");

    //see metadata, and look how many columns in the table.
    fread( DstBuf: buffer, ElementSize: PAGE_SIZE, Count: 1, File: fp);
    int num_col = buffer[0] - '0';

    std::string store[num_col];

    for(int i=1 ; i<=num_col ; i++){
        printf( format: "%d (st/nd/rd/th) column value ? :\n", i);
        std::cin >> store[i-1];
    }

    //find the first
    //fread

    std::string left =""; // 6byte
    std::string right = ""; // ?

    //jump = sees metadata and check the col is varchar ? or not ?
    int jump = num_col +1;
    for(int i=1 ; i<=num_col ; i++){
        jump +=(buffer[i]-'0');
    }

    //가변길이 레코드 포맷 varchar start point.,
    int start_varchar = 1;
    for(int i=1 ; i<=num_col; i++){
        if(buffer[jump+i] == '0'){
            //char
            start_varchar += (buffer[jump+i+num_col] - '0');
        }else{
            //varchar
            start_varchar += 3;
        }
    }

    for(int i=1; i<=num_col ;i++){
        if(buffer[jump+i] == '0'){
            right += store[i-1];
```

```
            right += store[i-1];
        }else{
            right += std::to_string( val: start_varchar);
            int length = store[i-1].length();
            right += std::to_string( val: length);
            start_varchar += length;
        }
    }

    right +=  "0"; // null bitmap

    for(int i=1 ; i<=num_col; i++){
        if(buffer[jump+i] =='1' ){
            right += store[i-1];
        }
    }
    //std::cout << right << std::endl; //00001M14241630cskim

    int p = 1;
    while(true) {
        rewind( File: fp);
        fseek( File: fp,  Offset: PAGE_SIZE * p,  Origin: SEEK_SET);
        fread( DstBuf: buffer,  ElementSize: ONE_LINE,  Count: 1,  File: fp);

        //std::cout << buffer <<std::endl;

        std::string now_buffer( s: buffer);
        if(avail_free_space( info: now_buffer,  len: right.length())){
            std::string sub_str = now_buffer.substr( pos: 3, n: 3);
            int end_free_space = std::stoi( str: sub_str);

            std::string sub_str2 = now_buffer.substr( pos: 0, n: 3);
            int start_free_space = (std::stoi( str: sub_str2)+1) * 6;

            left += make_three_letter( sample: std::to_string( val: right.length()));
            left += make_three_letter( sample: std::to_string( val: end_free_space - right.length()));

            now_buffer.replace( pos: start_free_space,  n: 6,  str: left);
            now_buffer.replace( pos: end_free_space - right.length() + 1,  n: right.length(),  str: right);
            now_buffer.replace( pos: 3, n: 3, str: std::to_string( val: end_free_space - right.length()));
            now_buffer.replace( pos: 0, n: 3 , str: make_three_letter( sample: std::to_string( val: std::stoi( str: sub_str2)+1)));

            rewind( File: fp);
            fseek( File: fp,  Offset: PAGE_SIZE * p,  Origin: SEEK_SET);
            fwrite( Str: now_buffer.c_str(),  Size: now_buffer.length(),  Count: 1,  File: fp);
            break;
```

```
    }

    p = p+1;
}
```

+추가자료) 지난 5월보고서에서 인용

첫 줄의 메타데이터는

526644idgenderdepartyearprof0010151010 정도가 되겠습니다(남은 길이는 , space로 채운다.)

5 / 26644/idgenderdepartyearprof/ 00101 / 511

5 -> 칼럼수가 다섯이며

26644 -> 각 칼럼 이름의 크기는 2,6,6,4,4 이다.

Id (2글자) -> 첫번째 칼럼 이름이 id다

Gender(6글자) -> 두번째 칼럼 이름이 gender이다.

Depart(6글자) -> 세번째 칼럼 이름이 depart 이다.

Year(4글자) -> 네번째 칼럼 이름이 year이다.

Prof(4글자) -> 다섯번째 칼럼 이름이 prof이다.

00101 -> 세번째 , 다섯번째 칼럼이 가변이고 나머지는 고정길이 이다.

51010 -> 첫번째 고정길이 칼럼 길이가 5이고, 두번째는 1, 네번째 1 이다.

가변 길이 레코드 포맷은 , 수업 때 했던 것과 완전히 동일 하게 할 생각입니다.

아까 table 에서 (1,m , computer_science, 4, kim) 이라는 record 가 있다고 할 때

00001(5byte)	M(1byte)	14,2 (3byte)	4(1byte)	16,3 (3byte)	Null bitmap (00000000)	Cs	kim

과 같이 되겠습니다.

*이 때 max column 수는 8 으로 하겠습니다. (null bitmap 추가확장 없음)

***1 page 크기를 300 bytes로 두어 ,** 약 10개내외의 record가 1 page에 들어가도록 할 의도입니다.

(1page 에 record 가 너무 많이 들어가면 정상적으로 동작하는지 체크가 어려워서 페이지 크기를 조금 작게 두고자 합니다.)

(3) 레코드 검색 & 컬럼 검색

레코드 검색은 각 블락 처음 3byte 을 읽어 몇 record가 들어있는 지 읽고

그만큼 for loop 돌며 id 를 검사 합니다. (못 찾으면 다음 block 을 읽습니다)

*다만 리코드가 아예 없을 시에는 block 의 record 수가 0으로 읽히니까 그때 for loop 빠져나와서 못찾았다는 출력을 합니다.

```
3. Search record & Search column
==================================
3
type table name!
student
type id that you want!
22
Cannot Find record.

Process finished with exit code 0
```

그 다음에 찾은 record 의 특정 column 검색을 진행할 건지 묻습니다.

진행하면, metadata를 읽어서 찾고자 하는 column 이 해당 record 안에서 얼만큼 offset 을 두고, 얼마의 크기만큼 읽을 지 계산합니다. (범위 계산과정은 코드 참조)

범위 계산이 다 되었으면 record의 일부를 읽어서 출력합니다.

```
==================================
3
type table name!
student
type id that you want!
4
Found Record! Record is  ::
00004F14411840MATHCHOI  .
Continue with searching column ? (yes = 1 , no = 0)
1
Input the column name that you want.
prof
column value is :: CHOI

Process finished with exit code 0
```

또한 null bitmap 을 수정 및 검사 해보겠습니다.

여기서 2번째 record의 2번째 칼럼(gender) 에 해당하는 null bitmap을 B로 만들었습니다.

(*B = 48 = 32+16 , 4번째 5번째 칼럼이 NULL BITMAP 이 1로 바뀐것임)

```
00002F1421163BEEKIM00001M14241630CSLEE
00004F14411840MATHCHOI00003M14231630EELEE
```

```
3. Search record & search column
==================================
3
type table name!
student
type id that you want!
2
Found Record! Record is  ::
00002F1421163BEEKIM  .
Continue with searching column ? (yes = 1 , no = 0)
1
Input the column name that you want.
year
column value is :: null

Process finished with exit code 0
```

```
==================================
3
type table name!
student
type id that you want!
2
Found Record! Record is  ::
00002F1421163BEEKIM  .
Continue with searching column ? (yes = 1 , no = 0)
1
Input the column name that you want.
prof
column value is :: null

Process finished with exit code 0
```

*코드

```
}else if(now_menu == 3){
    //레코드 검색
    printf( format: "type table name!\n");
    std::string now_finding;
    std::cin >> now_finding;
    std::string temp_directory = ".\\tables\\" + now_finding +".txt";
    fp = fopen( Filename: temp_directory.c_str(), Mode: "r+");

    printf( format: "type id that you want!\n");
    int find_id;
    scanf( format: "%d", &find_id);

    int p = 1;
    int find_flag = 0;
    while(true){
        rewind( File: fp);
        fseek( File: fp, Offset: PAGE_SIZE * p, Origin: SEEK_SET);
        fread( DstBuf: buffer, ElementSize: ONE_LINE, Count: 1, File: fp);
        std::string now_buffer( s: buffer);

        int n = std::stoi( str: now_buffer.substr( pos: 0, n: 3)); // how many records in that block?
        if(n==0) break;
        for(int i=1; i<=n ; i++) {
            int now_size = std::stoi( str: now_buffer.substr( pos: 6*i, n: 3));
            int now_start = std::stoi( str: now_buffer.substr( pos: 6*i+3, n: 3));
            //std::cout << now_buffer.substr(now_start+1, 5) << std::endl;
            int now_id = std::stoi( str: now_buffer.substr( pos: now_start+1, n: 5));

            if(now_id == find_id){
                prev_record = now_buffer.substr( pos: now_start+1, n: now_size);
                find_flag = 1;
                break;
            }
        }

        if(find_flag == 1) break;
        p = p+1;
    }
```

```
}

int flag_column_search = 0;
if(find_flag == 1) {
    std::cout << "Found Record! Record is  :: " << std::endl;
    std::cout << prev_record << "  ." << std::endl;
    std::cout << "Continue with searching column ? (yes = 1 , no = 0)" << std::endl;
    std::cin >> flag_column_search;
}else{
    std::cout << "Cannot Find record. " <<std::endl;
}

if(flag_column_search == 1){
    std::string search_col ;
    std::cout << "Input the column name that you want." << std::endl;
    std::cin >> search_col;

    rewind( File: fp);
    fread( DstBuf: buffer,  ElementSize: ONE_LINE,  Count: 1,  File: fp);
    std::string now_buffer( s: buffer);

    int n = std::stoi( str: now_buffer.substr( pos: 0, n: 1));

    int place = -1; // 몇번째  col 하고 일치 ?
    int p = n+1;
    for(int i=1 ; i<=n ; i++){
        int now_len = std::stoi( str: now_buffer.substr( pos: i, n: 1));
        if(now_len == search_col.length()){
            if(now_buffer.substr( pos: p,  n: now_len).compare( str: search_col) == 0){
                place = i;
                break;
            }
        }
        p+=now_len;
    }

    if(place == -1){
        std::cout << "you typed incorrect col name . " << std::endl;
    }else {
```

```
    //jump = sees metadata and check the col is varchar ? or not ?
    int jump = n+1;
    for(int i=1 ; i<=n; i++){
        jump +=(buffer[i]-'0');
    }

    int col_p = 0;
    for (int i = 1; i < place; i++) {
        if (buffer[jump + i] == '0') {
            //char
            col_p += (buffer[jump+n+i] - '0');
        } else {
            //varchar
            col_p += 3;
        }
    }
    //std::cout << prev_record << std::endl;
    std::string col_result ;
    if((buffer[jump] & (1<<place)) == 1){
        //null bitmap
        col_result = "null";
    }else{
        if(buffer[jump+place] == '0'){
            //char
            int temp = buffer[jump+n+place]-'0';z
            col_result = prev_record.substr(col_p, temp);
        }else{
            //varchar
            int start =std::stoi( str: prev_record.substr( pos: col_p, n: 2));
            int end =  std::stoi( str: prev_record.substr( pos: col_p+2, n: 1));
            col_result = prev_record.substr( pos: start ,  n: end);
        }
    }

    std::cout <<"column value is :: " <<col_result << std::endl;
}

}
```