Berkeley DB for TinyIoT

Sejong Univ.

Name : Park Minji
E-mail : iorw0224@gmail.com

Berkeley DB 이번주 진행 상황

- DB 정렬 기준
- Cursor 동작 원리
- Get_CIN_Period (start_time, end_time) 추가해 특정 범위에 해당하는 cin들을 검색할 수 있음


Github


Notion

Berkeley DB DB 정렬 원리

DB Key 정렬 기준
: 사전순(숫자-대문자-소문자 순서), 길이순

AE.db

Key	Value
aei	TAE1
aei	TAE3
aei	TAE2
api	tinyProject1
api	tinyProject3
api	tinyProject2
ct	20220513T083900
ct	20220513T083900
ct	20220513T083900
et	20240513T083900
et	20240513T083900
et	20240513T083900
lt	20220513T083900
lt	20220513T083900
lt	20220513T083900
pi	5-20191210093452845
pi	5-20191210093452845
pi	5-20191210093452845
ri	TAE1
ri	TAE3
ri	TAE2
rn	Sensor1
rn	Sensor3
rn	Sensor2
rr	true
rr	true
rr	true
ty	2
ty	2
ty	2

DB Value 정렬 기준
: 들어온 순서대로

Berkeley DB 특정 레코드를 검색할 때 커서 동작 원리

```
// 오브젝트가 몇개인지 찾기 위한 커서
DBC* dbcp0;
if ((ret = dbp->cursor(dbp, NULL, &dbcp0, 0)) != 0) {
    dbp->err(dbp, ret, "DB->cursor");
    exit(1);
}
while ((ret = dbcp0->get(dbcp0, &key, &data, DB_NEXT)) == 0) {
    if (strncmp(key.data, "ri", key.size) == 0) {
        cnt++; // 오브젝트 개수
    }
}
```

// 커서 생성

// 다음 레코드로 넘어가면서 탐색

3개의 오브젝트

arr

0	1	0

오브젝트 수 만큼 동적할당

arr

0	1	0

idx

찾을 오브젝트의 인덱스를 1로 바꿈

AE.db

Cursor →

순차적으로 레코드 탐색

Key	Value
aei	TAE1
aei	TAE3
aei	TAE2
api	tinyProject1
api	tinyProject3
api	tinyProject2
ct	20220513T083900
ct	20220513T083900
ct	20220513T083900
et	20240513T083900
et	20240513T083900
et	20240513T083900
lt	20220513T083900
lt	20220513T083900
lt	20220513T083900
pi	5-20191210093452845
pi	5-20191210093452845
pi	5-20191210093452845
ri	TAE1
ri	TAE3
ri	TAE2
rn	Sensor1
rn	Sensor3
rn	Sensor2
rr	true
rr	true
rr	true
ty	2
ty	2
ty	2

Berkeley DB 특정 레코드를 검색할 때 커서 동작 원리

```
while ((ret = dbcp->get(dbcp, &key, &data, DB_NEXT)) == 0) {
    if (strncmp(key.data, "pi", key.size) == 0) {
        if (arr[idx % cnt]) {
            //printf("[%d]", idx % cnt);
            node_pi->pi = malloc(data.size);
            strcpy(node_pi->pi, data.data);
            node_pi->siblingRight = (Node*)malloc(sizeof(Node));
            node_pi->siblingRight->siblingLeft = node_pi;
            node_pi = node_pi->siblingRight;
        }
        idx++;
    }
    if (strncmp(key.data, "ri", key.size) == 0) {
        if (arr[idx % cnt]) {
            node_ri->ri = malloc(data.size);
            strcpy(node_ri->ri, data.data);
            node_ri = node_ri->siblingRight;
        }
        idx++;
    }
    if (strncmp(key.data, "rn", key.size) == 0) {
        if (arr[idx % cnt]) {
            node_rn->rn = malloc(data.size);
            strcpy(node_rn->rn, data.data);
            node_rn = node_rn->siblingRight;
        }
        idx++;
    }
    if (strncmp(key.data, "ty", key.size) == 0) {
        if (arr[idx % cnt]) {
            node_ty->ty = *(int*)data.data;
            node_ty = node_ty->siblingRight;
        }
        idx++;
    }
```

레코드를 찾아 배열에 표시 했으면,
해당 인덱스를 이용해 오브젝트 반환

AE.db

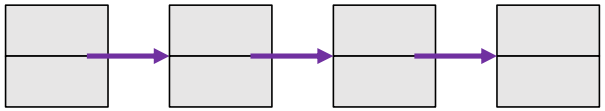
Key	Value
aei	TAE1
aei	TAE3
aei	TAE2
api	tinyProject1
api	tinyProject3
api	tinyProject2
ct	20220513T083900
ct	20220513T083900
ct	20220513T083900
et	20240513T083900
et	20240513T083900
et	20240513T083900
lt	20220513T083900
lt	20220513T083900
lt	20220513T083900
pi	5-20191210093452845
pi	5-20191210093452845
pi	5-20191210093452845
ri	TAE1
ri	TAE3
ri	TAE2
rn	Sensor1
rn	Sensor3
rn	Sensor2
rr	true
rr	true
rr	true
ty	2
ty	2
ty	2

Cursor → pi

Key = “pi”

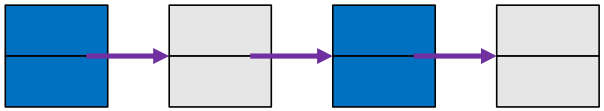
Berkeley DB 서버와 동작 방식

DB



해당 시간에 포함되는 cin들을 모두 Node 형태로 반환

Server



ri와 pi를 비교하여 유효한 cin만 추출

Berkeley DB Get_CIN_Period // 해당 시간에 포함되는 CIN들을 노드로 반환하는 함수

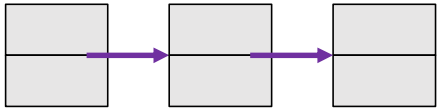
함수 원형

```
Node* Get_CIN_Period(char* start_time, char* end_time)
```

인자 : 검색할 시간 (Start time, End time)

```
<20220807T222120 ~ 20220808T224951>
```

반환형태 : Node



// 현재 시간을 아래 형태로 반환하는 함수

```
char* Get_LocalTime()
```

반환 값 : 20220807T222120

year month day hour sec min

'4-' 를 앞에 붙혀서 ri 형태로 만듦

Ri 형태: 4-20220807T222120

type year month day hour sec min

출력 결과 (2020-08-07, 22시 21분 20초 ~ 2020-08-08, 22시 49분 51초)

```
<20220807T222120 ~ 20220808T224951>
4-20220808T113154
4-20220808T093154
4-20220807T233154
```

Berkeley DB Get_CIN_Period // 해당 시간에 포함되는 CIN들을 노드로 반환하는 함수

CIN.db

Key	Value
ri	4-20220513T093154
ri	4-20220808T113154
ri	4-20220808T093154
ri	4-20220807T233154
ri	4-20220807T113154

범위에 해당하는 ri

Key	Value
ri	4-20220513T093154
ri	4-20220808T113154
ri	4-20220808T093154
ri	4-20220807T233154
ri	4-20220807T113154

Cursor →

// 해당하는 오브젝트 배열에 1로 표시
0 1 1 1 0 <- 두번째 세번째 네번째 오브젝트가 해당

```
while ((ret = dbcp1->get(dbcp1, &key, &data, DB_NEXT)) == 0) {
    if (strncmp(key.data, "ri", key.size) == 0) {
        //9 : date, 16: time
        if (strncmp(start_ri, data.data, 16) <= 0 && strncmp(end_ri, data.data, 16) >= 0)
            arr[idx] = 1;
        idx++;
    }
}
```

cin 개수(cnt) = 5

arr

0	1	1	1	0

idx

// 범위에 해당하는 오브젝트가 없을 때 -> NULL 반환

```
int sum = 0;
for (int i = 0; i < cnt; i++) {
    sum += arr[i];
}
if (sum == 0) {
    fprintf(stderr, "Data not exist₩n");
    return NULL;
    exit(1);
}
```

arr

0	0	0	0	0

sum =0