Sprawozdanie - Zaawansowane technologie bazodanowe

Laboratorium 4 - 06.12.2019

Lekkie technologie bazodanowe

Dominik Wróbel

Zadanie 1

W tym ćwiczeniu wykonano eksperyment polegający na alokacji pamięci dla 1 000 000 struktur i przypisania do ich pól wartości. Następnie w strukturach wyszukano element o id 999 999.

Zmierzono czas wykonania wypełniania i wyszukiwania, a także zużycie pamięci. Operacje te wykonywane są bez użycia bazy danych, pozwoli to na późniejsze porównanie do tych samych operacji wykonywanych na bazie danych.

Program wykonano na systemie operacyjnym Linux Ubuntu, działającym na wirtualnej maszynie.

Wyniki z wykonania programu:

```
dominik@dominik-VirtualBox:~/Desktop$ ./db1
FILLING time, c2-c1[s]: 0.028007
FINDING time, c4-c3[s]: 0.008681
Max mem usage[kB]: 114224
dominik@dominik-VirtualBox:~/Desktop$ ./db1
FILLING time, c2-c1[s]: 0.027894
FINDING time, c4-c3[s]: 0.008820
Max mem usage[kB]: 114348
dominik@dominik-VirtualBox:~/Desktop$ ./db1
FILLING time, c2-c1[s]: 0.025502
FINDING time, c4-c3[s]: 0.008871
Max mem usage[kB]: 114316
dominik@dominik-VirtualBox:~/Desktop$ ./db1
FILLING time, c2-c1[s]: 0.023931
FINDING time, c4-c3[s]: 0.008693
Max mem usage[kB]: 114312
```

Wyniki średnie:

	Czas wstawiania	Czas wyszukiwania	Zużycie pamięci
Średnia	0,0263335s	0,00876625s	114 300 kB

Kod źródłowy do zadania 1

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys/resource.h>

#define MAX 1000000

struct Rec {
    int id;
    char name[20];
    char desc[90];
};

struct rusage mem;

typedef struct Rec Rec;

int main() {


    Rec * data = (Rec *) malloc(sizeof(Rec)*MAX);
    if(data == NULL){
        fprintf(stderr, "Mem alloc error \n.");
        return 1;
    }

    // -------- FILLING DATA START
    clock_t c1,c2;

    c1=clock();
    int i;
    for(i = 0; i < MAX; i++){
        data[i].id = i;
        strcpy(data[i].name, "Name");
        strcpy(data[i].desc, "Desc");
    }

    c2=clock();
    printf("FILLING time, c2-c1[s]: %f\n", ((float)c2-(float)c1)/CLOCKS_PER_SEC);
    // --------- FILLING DATA END


    // --------- FINDING DATA START
    clock_t c3,c4;
    c3=clock();
    int j;
    for(j = 0; j < MAX; j++){
        if(data[j].id==999999){
            break;
        }
    }
    c4=clock();
    printf("FINDING time, c4-c3[s]: %f\n", ((float)c4-(float)c3)/CLOCKS_PER_SEC);
    // --------- FINDING DATA END


    // --------- MEMORY USAGE
    getrusage(RUSAGE_SELF,&mem);
    printf("Max mem usage[kB]: %ld\n", mem.ru_maxrss);

    free(data);

    return 0;
}
```

Zadanie 2

W tym ćwiczeniu powtórzono pomiary z zadania pierwszego, ale tym razem dane są zapisywane w bazie danych sqlite. Eksperymenty prowadzone są z uwzględnieniem kilku parametrów i różnych ich kombinacji:

	Transakcje	Baza danych	Id	Name	Desc
Wariant I	Nie	Plik	Int	Char(8)	Char(15)
Wariant II	Nie	Plik	Int	Char(18)	Char(80)
Wariant III	Nie	Pamięć	Int	Char(8)	Char(15)
Wariant IV	Nie	Pamięć	Int	Char(18)	Char(80)
Wariant V	Tak	Plik	Int	Char(8)	Char(15)
Wariant VI	Tak	Plik	Int	Char(18)	Char(80)
Wariant VII	Tak	Pamięć	Int	Char(8)	Char(15)
Wariant VIII	Tak	Pamięć	Int	Char(18)	Char(80)
Wariant 0	Brak bazy danych				

Porównanie średnich wyników eksperymentów dla każdego z wariantów:

Średnia pomiarowa	Czas wstawiania [s]	Czas wyszukiwania [s]	Zużycie pamięci [kB]
Wariant I	> 120s	-	-
Wariant II	> 120s	-	-
Wariant III	35,782310	0.000076	128716
Wariant IV	38,625805	0,000088	349672
Wariant V	30.327076	0.000592	98688
Wariant VI	32.355591	0.000528	207900
Wariant VII	29.851763	0.000088	144740
Wariant VIII	31.531944	0.000108	376068
Wariant 0	0,0263335s	0,00876625s	114 300 kB

Wnioski:

- Odczyt z bazy przechowywanej w pamięci jest szybszy
- Zapis do bazy przechowywanej w pamięci jest szybszy
- Zapis i odczyt większych zmiennych jest wolniejszy
- Transakcje mają duży wpływ na czas działania w przypadku wielu operacji

Wyniki działa programu:

- Wariant III

```
dominik@dominik-VirtualBox:~/Desktop$ ./db2
FILLING time, c2-c1[s]: 35.782310
I found sth: NameName, DescDescDescDes, 999999
FINDING time, c4-c3[s]: 0.000076
Max mem usage[kB]: 128716
```

- Wariant IV

```
dominik@dominik-VirtualBox:~/Desktop$ ./db2
FILLING time, c2-c1[s]: 38.625805
I found sth: NameNameNameNameNa, DescDescDescDesDescDescDescDesDescDescDescDesDe
scDescDescDesDescDescDescDesDesc, 999999
FINDING time, c4-c3[s]: 0.000088
Max mem usage[kB]: 349672
```

- Wariant V

```
dominik@dominik-VirtualBox:~/Desktop$ ./db2
FILLING time, c2-c1[s]: 30.287643
I found sth: NameName, DescDescDescDescDes, 999999
FINDING time, c4-c3[s]: 0.000412
Max mem usage[kB]: 98688
```

- Wariant VI

```
dominik@dominik-VirtualBox:~/Desktop$ ./db2
FILLING time, c2-c1[s]: 32.355591
I found sth: NameNameNameName, DescDescDescDescDesDescDescDescDesDescDescDes
cDescDesDescDescDescDescDesDescDescDescDesDescc, 999999
FINDING time, c4-c3[s]: 0.000528
Max mem usage[kB]: 207900
```

- Wariant VII

```
dominik@dominik-VirtualBox:~/Desktop$ ./db2
FILLING time, c2-c1[s]: 29.851763
I found sth: NameName, DescDescDescDescDes, 999999
FINDING time, c4-c3[s]: 0.000088
Max mem usage[kB]: 144740
```

- Wariant VIII

```
dominik@dominik-VirtualBox:~/Desktop$ ./db2
FILLING time, c2-c1[s]: 31.531944
I found sth: NameNameNameName, DescDescDescDescDesDescDescDescDesDescDescDes
cDescDesDescDescDescDescDesDescDescDescDesDescc, 999999
FINDING time, c4-c3[s]: 0.000108
Max mem usage[kB]: 376068
```

Kod źródłowy do zadania 2

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys/resource.h>
#include <string.h>
#include <sqlite3.h>

#define MAX 1000000

struct Rec {
    int id;
    char name[20];
    char desc[90];
};

struct rusage mem;

typedef struct Rec Rec;

int main() {

    int rc;
    sqlite3 * db;
    // storage option:
    // char loc[] = ":memory:"; // in memory
    char loc[] = "./db.sqlite"; // file

    // open db ------------------------------------ //
    rc=sqlite3_open(loc,&db);
    if (rc){
        fprintf(stderr, "Can't open database: %s\n", sqlite3_errmsg(db));
        return 1;
    }

    // create table ------------------------------- //
    rc=sqlite3_exec(db,"CREATE TABLE inv (id integer PRIMARY KEY, name varchar(20), desc varchar(90));", NULL, NULL, NULL);

    if (rc){
            fprintf(stderr, "Database create table error: %s\n", sqlite3_errmsg(db));
            return 1;
    }

    // insert data -------------------------------- //
    clock_t c1 = clock();
    sqlite3_exec(db, "BEGIN TRANSACTION", NULL, NULL, NULL);
    int i;
    for(i = 0; i < MAX; i++){
        char * sqlQuery = sqlite3_mprintf("INSERT INTO inv VALUES(%d, 'NameNameNameName', 'DescDescDescDescDesDescDescDescDescDesDescDescDescDescDesDescDescDescDescDesDescc');", i);
        int rc = sqlite3_exec(db, sqlQuery, NULL, NULL, NULL);
        if(rc) {
            fprintf(stderr, "Insert error: %s\n", sqlite3_errmsg(db));
        }
    }
    sqlite3_exec(db, "COMMIT TRANSACTION", NULL, NULL, NULL );
    clock_t c2 = clock();
    printf("FILLING time, c2-c1[s]: %f\n", ((float)c2-(float)c1)/CLOCKS_PER_SEC);

    // find data with id -------------------------- //
```

```
60.     clock_t c3 = clock();
61.     int idWanted = 999999;
62.     char * sqlQuery = sqlite3_mprintf("SELECT * from inv WHERE id=%d;", idWanted);
63.
64.     sqlite3_stmt *stmt;
65.     rc=sqlite3_prepare_v2(db, sqlQuery, strlen(sqlQuery), &stmt, NULL);
66.
67.     if (rc) {
68.         fprintf(stderr, "Database prepare statement error: %s\n", sqlite3_errmsg(db)
    );
69.         return 1;
70.     }
71.     do {
72.         rc=sqlite3_step(stmt);
73.     if (rc==SQLITE_ROW) {
74.         printf("I found sth: %s, %s, %d\n",
75.             (char *)sqlite3_column_text(stmt,1),
76.             (char *)sqlite3_column_text(stmt,2),
77.             sqlite3_column_int(stmt,0));
78.     }
79.     } while (rc == SQLITE_ROW);
80.     sqlite3_finalize(stmt);
81.     clock_t c4 = clock();
82.
83.     printf("FINDING time, c4-c3[s]: %f\n", ((float)c4-(float)c3)/CLOCKS_PER_SEC);
84.
85.     // memory usage ------------------------------ //
86.     getrusage(RUSAGE_SELF,&mem);
87.     printf("Max mem usage[kB]: %ld\n", mem.ru_maxrss);
88.
89.
90.     // close db ---------------------------------- //
91.     sqlite3_close(db);
92. }
```