Projeto de Laboratórios de Informática 3
Grupo 20

Diogo Miguel Alves Rocha (A79751) Gabriela Sá Martins (A81987)
Ricardo Milhazes Veloso (A81919) Ricardo Jorge Silva Ferreira (82568)

5 de Maio de 2018

Resumo

O objetivo deste trabalho consistiu em criar um programa que permitisse a extração e análise de dados provenientes da base de dados correspondente ao StackOverFlow. Tendo como objetivo final a resposta a um determinado número de querys , em que o tempo de execução será o mais reduzido possível.

Conteúdo

1 Introdução **2**

2 Descrição do Problema **2**

3 Concepção da Solução **2**
3.1 Porquê da HashTable 2

4 A Estrutura **2**
4.1 h_IDUtilizador 3
4.2 h_IDPerguntas 3
4.3 h_IDRespostas 4
4.4 h_IDTags 4

5 Extração de Informação **4**
5.1 Pergunta 1 4
5.2 Perguntas 2,3,4,5,6,7,8,9,10,11 4
5.2.1 Pergunta 2 5
5.2.2 Pergunta 3 5
5.2.3 Pergunta 4 5
5.2.4 Pergunta 5 5
5.2.5 Pergunta 6 5
5.2.6 Pergunta 7 5
5.2.7 Pergunta 8 5
5.2.8 Pergunta 9 5
5.2.9 Pergunta 10 5
5.2.10 Pergunta 11 6

6 Resultados **6**

7 Teste de Tempo **6**

8 Memory Leaks **6**

9 Conclusão 7

1 Introdução

O trabalho proposto tinha em vista a resolução de 11 interrogações da forma mais eficiente e rápida possível. Para que conseguissemos corresponder a estas medidas criamos quatro estruturas, todas elas HashTables .

2 Descrição do Problema

O principal problema era criar uma estrutura capaz de armazenar todos os dados provenientes da base de dados correspondente ao StackOverFlow, de maneira que o acesso a esses mesmos dados fosse o mais rápido possível que por consequência a resposta às interrogações fosse ainda mais rápida, e sem memory leaks.

3 Concepção da Solução

Com vista a resolver o problema decidimos criar 4 HashTables, uma com os dados do utilizador,**h_IDUtilizador**,outra com os dados referentes aos posts do tipo pergunta **h_IDPerguntas**,outra com os dados referentes aos posts do tipo resposta **h_IDRespostas**, e por fim outra com as tags **h_IDTags**. Cada uma das anteriores será explicitada numa secção posterior referente às mesmas.

3.1 Porquê da HashTable

O principal motivo da escolha deste tipo de estrutura de dados foi o tempo de procura de um elemento que no melhor caso vai ser sempre constante e é o tipo de estrutura que apresenta o tempo de procura inferior em relação a outros.

Data Structure	Time Complexity			
	Average			
	Access	Search	Insertion	Deletion
Array	O(1)	O(n)	O(n)	O(n)
Stack	O(n)	O(n)	O(1)	O(1)
Queue	O(n)	O(n)	O(1)	O(1)
Singly-Linked List	O(n)	O(n)	O(1)	O(1)
Doubly-Linked List	O(n)	O(n)	O(1)	O(1)
Skip List	O(log(n))	O(log(n))	O(log(n))	O(log(n))
Hash Table	N/A	O(1)	O(1)	O(1)
Binary Search Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))
Cartesian Tree	N/A	O(log(n))	O(log(n))	O(log(n))
B-Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))
Red-Black Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))
Splay Tree	N/A	O(log(n))	O(log(n))	O(log(n))
AVL Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))
KD Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))

Figura 1: timecomplexity

4 A Estrutura

```
struct TCD_community
{
    GHashTable *h_IDUtilizador;
    GHashTable *h_IDPerguntas;
    GHashTable *h_IDRespostas;
    GHashTable *h_IDTags;
```

Figura 2: TCD_community

4.1 h_IDUtilizador

A estrutura contém informações relativas ao **utilizador**, e apresenta 7 parâmetros diferentes:

id_u -ID do utilizador;

name -Nome do utilizador;

shortb -Short bio do utilizador;

totalposts -Número total de posts;

reputacao -Reputação do utilizador;

votes -Número de votes do utilizador;

```
struct h_IDUtilizador
{
    long id_u;
    char* name;
    char* shortb;
    int totalposts;
    long reputacao;
    int votes;
};
```

Figura 3: h_IDUtilizador

4.2 h_IDPerguntas

A estrutura contém informações relativas às **Perguntas**, e apresenta 7 parâmetros diferentes:

id_post -ID da pergunta;

id_autor -ID do autor da pergunta;

title -Título da pergunta;

tags -Tag da pergunta;

reputacao -Reputação do utilizador;

n_resp -Número de respostas àquela pergunta;

d -Data da pergunta;

```
struct h_IDPerguntas
{
    long id_post;
    long id_autor;
    char* title;
    char* tags;
    long score;
    long n_resp;
    char* d;
};
```

Figura 4: h_IDPerguntas

4.3 h_IDRespostas

A estrutura contém informações relativas às **Respostas**, e apresenta 6 parâmetros diferentes:

id_post -ID da Resposta;

id_perg -ID da pergunta correspondente à resposta;

id_autor -ID do autor da resposta;

score -Score da resposta;

n_coments -Número de comentários aquela resposta;

d -Data da Resposta;

```
struct h_IDRespostas
{
    long id_post;
    long id_perg;
    long id_autor;
    long score;
    long n_coments;
    char* d;
};
```

Figura 5: h_IDRespostas

4.4 h_IDTags

A estrutura contém informações relativas às **Tags**, e apresenta 3 parâmetros diferentes:

id_tag -ID da tag;

tag -Tag;

count -Conta o número de vezes que uma Tag é utilizada num intervalo de tempo;

```
struct h_IDTags
{
    long id_tag;
    char* tag;
    long count;
};
```

Figura 6: h_IDTags

5 Extração de Informação

5.1 Pergunta 1

Na pergunta 1 temos de retornar um par com o titulo do posto correspondente ao id recebido, e com o nome do utilizador a que esse post pertence. A função utilizada é a **g_hash_table_lookup.**

5.2 Perguntas 2,3,4,5,6,7,8,9,10,11

As seguintes perguntas foram obtidas usando a mesma função **getNextIterator**.

5.2.1 Pergunta 2

Nesta pergunta queremos saber o Top N utilizadores com maior número de posts e para isso consultamos a variável totalposts.

5.2.2 Pergunta 3

Nesta pergunta queremos obter o número total de posts num determinado periodo de tempo mas com perguntas e respostas em separado. Para isso depois de consultar a data em cada uma das hashatbles e comparar com a que é estipulada consultamos a variável id_post tanto na hashtable de perguntas como a de respostas. Returnado um par com o respetivo número de perguntas e respostas total.

5.2.3 Pergunta 4

Nesta pergunta queremos obter o número total de perguntas contendo uma determinada tag num determinado periodo de tempo, e retornar essas perguntas por cronologia inversa. Para isso depois de consultar a data em cada uma das hashatbles e comparar com a que é estipulada consultamos a variável tags , e retornamos uma lista com o id dessas mesmas tags.

5.2.4 Pergunta 5

Nesta pergunta queremos devolver a informação do perfil do utilizador (short bio) e os IDs dos seus 10 últimos posts (perguntas ou respostas),ordenados por cronologia inversa. Para isso devolvemos uma struct userinfo que contém a variável short_bio e ultimos10;

5.2.5 Pergunta 6

Nesta pergunta queremos saber, num intervalo de tempo arbitrario, quais as N respostas com mais votos, devolvendo uma lista com a variável id_post.

5.2.6 Pergunta 7

Nesta pergunta o objetivo é dado um intervalo de tempo arbitrário devolver os id's das N perguntas com mais respostas. Por ordem decrescente do número de respostas. Devolvendo uma lista com o Top N das variaveis Id_post.

5.2.7 Pergunta 8

Nesta pergunta o objetivo é dado uma palavra, devolver uma lista com os id's de n perguntas cujos titulos a contenham, ordenados por cronologia inversa. para responder utilizamos a variável id_post, devolvendo uma lista destas.

5.2.8 Pergunta 9

Nesta pergunta recebendo o ID de dois utilizadores devolver as últimas N perguntas que em que participaram esses dois utilizadores. Para isso é devolvida uma lista com a variável ultimasN;

5.2.9 Pergunta 10

Nesta pergunta recebendo o ID de uma pergunta obter a melhor resposta. Para isso retornamos a variável id_melhor.

5.2.10 Pergunta 11

Nesta pergunta recebendo um intervalo de tempo arbitrário devolver os ids das N tags mais usadas peços N utilizadores com maior reputação. Para isso retornamos uma lisata com a variável user_best_rep;

6 Resultados

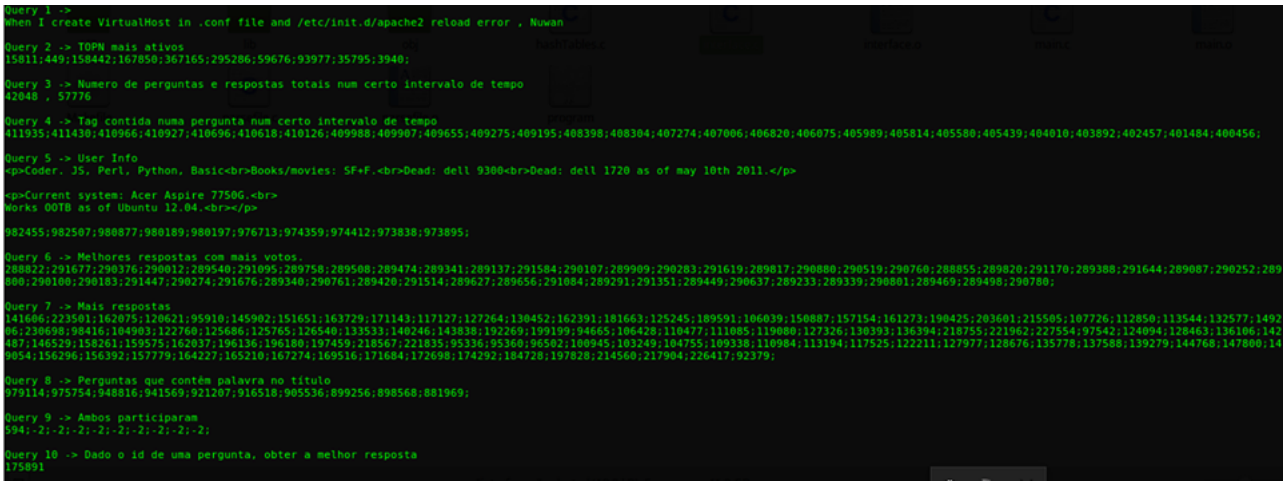

Figura 7: Resultados

7 Teste de Tempo

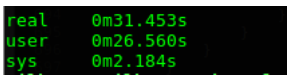

Figura 8: tempo

8 Memory Leaks

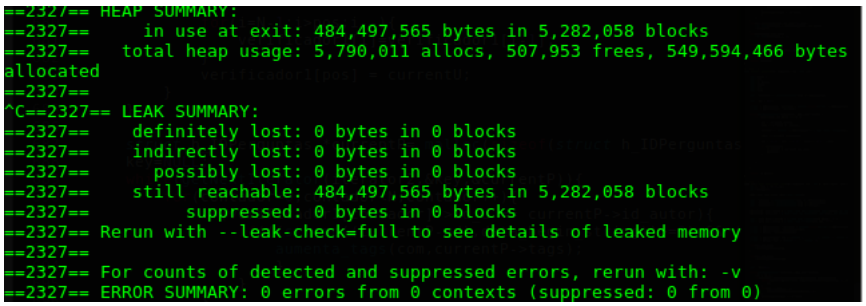

Figura 9: leaks utilizando o programa Valgrind

9 Conclusão

Em suma todos os objetivos propostos foram atingidos. Toda a extração de informação proveniente da base de dados do site **StackOverflow** e análise da mesma é feita de uma maneira eficiente, com um tempo de execução bastante aceitável e sem qualquer perda de informação.