Relatório do Projeto em *C*

Pedro Gonçalves (a82313) Rodolfo Silva (a81716)

Maio 2018

Universidade do Minho
Laboratórios de Informática 3
Grupo 47

Conteúdo

1 Introdução

Este relatório aborda a resolução do projeto prático em C de LI3. O projeto consiste em construir um sistema que permita analisar dados presentes em backups do Stack Overflow, e extrair informação útil desses dados como, por exemplo, o número de publicações num dado período de tempo, o número de novos utilizadores, entre outros.

Para ajudar nesta tarefa, foi necessário fazer o *parse* dos ficheiros e também criámos quatro módulos. Estes módulos são: o *parse* dos ficheiros dados, as estruturas de dados, código auxiliar para executar corretamente as queries fornecidas pelos professores, e um *program* para teste.

Depois dos ficheiros serem carregados, somos capazes de executar as queries disponibilizadas pela equipa docente. Para responder às diferentes queries utilizam-se as funções definidas nas API dos diferentes módulos já referidos.

Ao longo deste relatório abordam-se assim as decisões tomadas na implementação do projeto, nomeadamente quais as estruturas utilizadas para criar cada um dos módulos, o porquê das escolhas feitas e as suas APIs.

2 Módulos

Nesta secção apresentam-se excertos do código comentados, bem como explicações das soluções usadas e os 3 grupos em que dividimos o projecto.

2.1 Estrutura de Dados (*struct*)

Para armazenar toda a informação necessária à resolução das querys, optamos por usar três HashTables, uma para os utilizadores, outra para as publicações e outra para as tags. Foram usadas também duas ABin que continham outras informações sobre os utilizadores e sobre os posts. Decidimos utlizar estes métodos pois apesar de o carregamento de dados ser mais lento, a procura de informação é mais rápida.

2.1.1 TCD_community

A nossa TCD_community tem cinco parâmetros, sendo estes duas struct para os utilizadores, duas para as publicações e uma outra para as tags.

A primeira struct dos utilizadores, *USRP*, possui o id do utilizador, a sua reputação, a sua bio e por fim o seu nome de utilizador.

A segunda struct dos utilizadores, *UxPI*, possui o número de publicações do utilizador, tanto perguntas como respostas, o id dessas publicações, as datas em que essas publicações ocorreram, as perguntas em que o utilizador participou (tanto em pergunta como em resposta) e as tags utilizadas nessas publicações.

A primeira struct relativa às publicações, *PSTI*, contém o id da publicação, o id do utilizador que a criou (-2 se quem criou a publicação não possuía id), o tipo de publicação (pergunta ou resposta), o id da publicação mãe (-2, se a publicação for uma pergunta), o score da publicação, o número de comentários, o título (apenas se for pergunta), as tags usadas (apenos possui tags se for uma pergunta) e por fim a data de criação.

A segunda struct relativa às publicações, *PxUI*, contém o número de respostas a uma dada pergunta, um GArray contendo o id das respostas, um GArray contendo a data de publicação dessas mesmas respostas e por fim os id's dos utilizadores que criaram essas respostas.

A struct sobre as tags contém apenas o id referente a cada tag.

```
typedef struct TCD_community
{
  GTree *treePosts; // ABin com a struct PSTI
  GTree *treeUsers; // ABin com a struct USRP
  GHashTable *userXinfo; // HashTable com a struct UxPI
  GHashTable *infoXuser; // HashTable com a struct PxUI
  GHashTable *tagInfo; // HashTable com a struct TI
}TCD_community;
```

2.1.2 USRP

A *USRP struct* é uma ABin criada para os utilizadores. A key para a procura nesta ABin é o id dos utilizadores.

```c
typedef struct usr_posts {
   long usrID; // Id do utilizador
   int reputation;
   char* bio; // Bio do utilizador
   char* usrName; // Nome do utilizador
}usr_posts;
```

2.1.3 UxPI

A key para a procura nesta HashTable é o id dos utilizadores.

```c
typedef struct usrXpost_info {
  int post_total; // Numero total de posts
  GArray *postsID; // Id dos posts
  GArray *datas; // Data de publicacao dos posts
  GArray *questID; // Array com as perguntas em que cada user
      participou
  GArray *tags; // Array com as tags de cada publicacao
}usrXpost_info;
```

2.1.4 PSTI

A key para a prcoura nesta ABIN é o id dos posts.

```c
typedef struct post_Information {
   int postID; //ID do post
   int pUsrID; //ID do user que publicou
   int pType; //Tipo de Post (1==Pergunta || 2==Resposta)
   int parentID; //ID da pergunta mae (Apenas Resposta)
   int score; //Score do post
   int comments; //Numero total de comments do post
   char* titulo;
   char* tags;
   int dataNumber; //Data do post Concatenada
}post_Information;
```

2.1.5 PxUI

A key para a procura nesta HashTable é o id dos posts

```c
typedef struct postXusr_info {
  int dateP; // Data da pergunta
  int count; // Contador de numeros de elementos dos arrays
  GArray *answer; // Array com os ids das respostas
  GArray *usrID; // Id utilizadores que responderam
  GArray *datas; // Data publicaco respostas
}postXusr_info;
```

2.1.6 TI

A key para a procura nesta HashTable é o nome da tag.

```c
typedef struct tag_Info {
  long id; //Id da tag
}tag_Info;
```

2.2 Queries

Nesta secção mencionamos o modo como decidimos executar as queries fornecidas e de que maneira chegamos aos resultados fornecidos pela equipa docente.

2.2.1 Init

Nesta query apenas inicializamos a estrutura TAD_community, nenhum dos componentes da estrutura foi inicializado, pois estes serão inicializados nas funções parse que se faz na função load;

2.2.2 Load

Esta query possui a maior carga de trabalho do programa. São chamadas as funções parse das diferentes estruturas, e consequentemente o carregamento de dados para a estrutura.

2.2.3 Info From Post

Para a resolução desta query foram usadas as estruturas *PSTI* e *USRP*, e através de funções de procura em Abin do Glib, foi-se procurar pelo ID do post, de seguida verificou-se qual o tipo do post "pType".

Caso fosse uma Pergunta, retirou-se o titulo do post e o ID de quem publicou o post, de seguida procurou-se pelo seu ID na estrutura *USRP*, a partir da qual se tirou o nome do user a qual esse ID correspondia.

Caso fosse uma Resposta, retirou-se o "parentID"do post, que corresponde ao ID da pergunta a qual esta resposta foi dada, de seguida voltou-se ao passo 1 de procurar na estrutura *PSTI* e consecutivamente na estrutura *USRP*.

2.2.4 Top Most Active

Para a resolução desta query usamos a estrutura *UXPI* e através de funções de procura em HashTable do Glib, percorremos essa mesma HashTable e colocamos os ID dos Users e o número total de posts feitos por eles (post_total) foi usado como key para poder inserilos de maneira ordenada dentro de uma GSList que depois foi convertida para LONG_list.

2.2.5 Total Posts

Para a resolução desta query usamos a estrutura *PSTI*, e através das funções de travessia de Abin's do Glib,percorremos a Abin que continha essa mesma estrutura, verificando se a data do post, "dataNumber", estava dentro das datas fornecidas. Se estivesse, verificaríamos o tipo do post "pType"e dependendo do tipo incrementamos o contador desse mesmo tipo.

2.2.6 Questions With Tag

Para a resolução desta Query foi usada a estrutura *PSTI*, e atraves das funções de travessia de Abin's do Glib, percorremos a Abin que continha essa estrutura, onde verificamos se a data do post pertencia ao intervalo fornecido. Se isto se verificasse, verificamos se a tag pertencia ao post, caso ela pertencesse, retiramos o id da pergunta e a data da sua publicação, usando essa data como key para ordenar uma GSList com os id's que mais tarde seria convertida para uma LONG_list.

2.2.7 Get User Info

Para executar esta query, acedemos à ABin dos utilizadores utilizando o id fornecido na query como key, e retiramos daí os posts em que esse utilizador participou, e as datas em que esses posts foram publicados. Depois de obtida esta informação criamos uma GSList para ordenar os últimos dez posts feitos pelo utilizador. Depois de esta GSList estar criada criamos um array com esses 10 últimos posts e criamos um USER com esse array e com a bio do utilizador.

2.2.8 Most Voted Answers

Para a resolução desta query foi usada a estrutura PSTI, e através de funções de travessia de ABin's do Glib, percorremos as Abin que continha essa estrutura, onde testamos se a data do post pertencia ao intervalo fornecido. Se sim, retiramos o id das respostas e o seu score, usando o score para ordenar uma GSList que mais tarde seria convertida para uma LONG_list.

2.2.9 Most Answered Questions

Para a resolução desta query utilizamos a estrutura *PxUI*, e através de funções de travessia de HashTables's fornecidas pelo Glib, percorremos a HashTable dessa estrutura e verificavamos se as perguntas tinham sido feitas dentro do intervalo fornecido. Caso fosse verdadeiro, percorríamos as respostas relativas a esta pergunta e contávamos quantos dessas ocorriam dentro do intervalo de tempo dado. Por fim usávamos esse contador para ordenar uma GSList com os id's das N perguntas com mais respostas, sendo esta GSList mais tarde convertida para uma LONG_list.

2.2.10 Contains Word

Para a resolução desta query usamos a estrutura *PSTI*, e através das funções de travessia em Abin's do Glib, percorremos a Abin que continha essa estrutura, onde para cada nodo, íamos verificando se o título do post continha a palavra fornecida. Se a palavra ocorresse no título do post, retiravamos o id da pergunta e a data da sua publicação, usando a data da publicação para ordenar uma GSList que mais tarde seria convertida para uma LONG_list.

2.2.11 Both Participated

Para a resolução desta query utilizamos a estrutura *UxPI*, e através de funções de procura em Abin's fornecidas pelo Glib, retiramos os Garray's com as perguntas em que ambos tinham participado. Depois de obtermos esses Garray's víamos quais os posts em comum e preenchiamos uma GSlist ordena pelas datas das publicações em comum aos dois Id's. Depois convertíamos essa GSList para uma LONG_list.

2.2.12 Better Answer

Para a resolução desta query foram usadas as estruturas *PSTI*, *PxUI* e *USRP*, e através das funções de procura de Abin's e HashTable's do Glib, pesquisamos o ID do post na estrutura PxUI, de onde retiramos um GArray com os id's de todas as respostas a esse post. De seguida procuramos por esses id's na estrutura PSTI de onde retiramos o Score, Número de Comentários, e o id do utilizador que o fez essa resposta. Depois fomos procurar na estrutura USRP pelo user e tiramos a sua reputação.

Por fim usamos a reputação, score e número de comentários para calcular a média, que foi usada para comparar com a média mais alta obtida até aquele momento, caso fosse maior, essa passava a ser a maior, no final da função retornou-se o ID da resposta que tinha a maior média.

2.2.13 Clean

Nesta função usamos as funçôes de destruição fornecidas pelo Glib para dar free às diferentes HashTable's e Abin's usadas para armazenar as estruturas.

3 Modularidade

Atendendo à dimensão do projeto, foi necessária uma organização cuidada do código para que este seja controlável e legível, em todas as suas fases (desenvolvimento, teste e manutenção). O projeto foi dividido em 3 partes essenciais: load, init e parse. No load os dados são carregados para a estrutura, no init as estruturas são inicializadas e, por fim, no parse é feito o parsing dos dados através da biblioteca xmlib.

4 Melhora de Desempenho

Apesar de as querys implementadas responderem aos objetivos lançados pelos professores, é evidente que muito mais poderia ser feito para melhorar o desempenho geral do projecto, tais como, aprimorar o parse e o load de maneira a carregar quantidades inferiores de informação para as estruturas e melhorar as estruturas de dados de maneira a responder de forma ainda mais eficiente às querys. Existem portanto diversas formas de continuar o trabalho até aqui desenvolvido, todas elas revestidas de utilidade prática.

5 Conclusão

Em suma, concluimos o projeto com relativo sucesso. Através do parse dos ficheiros xml conseguimos extrair toda a informação relevante para o projeto, passando esta para uma estrutura de dados pensada com o obletivo de reponder às querys com a maior eficiência. Tínhamos como objetivo responder de maneira precisa e eficaz a todas as querys, e consoante os resultados pensamos que atingimos esse objectivo.

No entanto ficamos desapontados por não conseguirmos dar resposta à query11, apesar dos nossos esforços para a resolver.