Projeto de Laboratórios de Informática III
Grupo 25

Alexandre Mendonça Pinho (a82441) Joel Filipe Esteves Gama (a82202)
Tiago Martins Pinheiro (a82491)

12 de Junho de 2018

Conteúdo

Capítulo 1

Introdução

Este relatório apresenta uma explicação da segunda fase do projeto da disciplina de Laboratórios de Informática III, do 2º ano do Mestrado Integrado em Engenharia Informática da Universidade do Minho, que toma a forma de um projeto a ser desenvolvido na linguagem de programação *Java*.

Como projeto de avaliação é nos proposto responder a onze interrogações sobre processamento de dados de comunidades do Stack Exchange. Grande parte desta segunda parte do projeto consistiu em "converter"o código já feito na primeira fase do projeto para a nova linguagem de programação, até porque esta foi já feita com esta possibilidade em mente, criando classes para as estruturas de dados em ficheiros separados. No entanto, a segunda fase do projeto foi também uma oportunidade de melhorar a estrutura e implementação do que nos foi pedido.

Neste relatório apresentamos a descrição do problema a resolver e explicamos as estruturas utilizadas e os algoritmos de resposta às queries. Apresentamos também o raciocínio por detrás das decisões tomadas, bem como a forma como melhoramos os algoritmos.

Capítulo 2

Descrição do Problema

Interrogação 1: Dado o identificador de um post, a função deve retor- nar o título do post e o nome (não o ID) de utilizador do autor. Se o post for uma resposta, a função deverá retornar informações (título e utilizador) da pergunta correspondente;

Interrogação 2: Pretende obter o top N utilizadores com maior número de posts de sempre. Para isto, devem ser considerados tanto perguntas quanto respostas dadas pelo respectivo utilizador;

Interrogação 3: Dado um intervalo de tempo 4 arbitrário, obter o número total de posts (identificando perguntas e respostas separadamente) neste período;

Interrogação 4: Dado um intervalo de tempo arbitrário, retornar todas as perguntas contendo uma determinada tag. O retorno da função deverá ser uma lista com os IDs das perguntas ordenadas em cronologia inversa 5 ;

Interrogação 5: Dado um ID de utilizador, devolver a informação do seu perfil (short bio) e os IDs dos seus 10 últimos posts (perguntas ou res- postas), ordenados por cronologia inversa;

Interrogação 6: Dado um intervalo de tempo arbitrário, devolver os IDs das N respostas com mais votos, em ordem decrescente do número de votos;

Interrogação 7: Dado um intervalo de tempo arbitrário, devolver as IDs das N perguntas com mais respostas, em ordem decrescente do número de respostas;

Interrogação 8: Dado uma palavra, devolver uma lista com os IDs de N perguntas cujos títulos a contenham, ordenados por cronologia inversa;

Interrogação 9: Dados os IDs de dois utilizadores, devolver as últimas N perguntas (cronologia inversa) em que participaram dois utilizadores es- pecíficos. Note que os utilizadores podem ter participado via pergunta ou respostas;

Interrogação 10: Dado o ID de uma pergunta, obter a melhor resposta. Para isso, deverá usar a função de média ponderada abaixo: (Scr * 0.65) + (Rep * 0.25) + (Comt * 0.1) onde, - Scr - score da resposta; - Rep - reputação do utilizador; - Comt - número de comentários recebidos pela resposta;

Interrogação 11: Dado um intervalo arbitrário de tempo, devolver os identificadores das N tags mais usadas pelos N utilizadores com melhor re- putação. Em ordem decrescente do número de vezes em que a tag foi usada.

Capítulo 3

Conceção da Solução

3.1 Tipos de dados

Para representar a informação presente no ficheiros xml, foram codificadas três classes principais, cada uma com vários campos:

- User: id, reputação, nome de display, a descrição de utilizador, e o número de posts do utilizador;
- Post: id, id do post parente[1], id do utilizador que criou o post, pontuação, número de comentários, data de criação, tipo de post (pergunta, resposta, ou outro), flag de resposta aceite[2], nome de display do utilizador que criou o post (caso este já não exista), título[3] e o conjunto de respostas[4];
- Tag: id e nome

3.2 Estruturas de dados

Nesta fase do projeto utilizamos várias estruturas de dados para conseguir um melhor desempenho na reposta das queries individuais, em troca de fazer mais processamento ao fazer o parsing dos ficheiros:

- HashMap de ids de posts para posts;
- HashMap de ids de users para users;
- HashMap de nomes de tags para tags;
- TreeSet de posts ordenados cronologicamente;
- HashMap de ids de users para o set ordenado cronologicamente dos ids dos seus posts;
- TreeSet de users ordenados por reputação;
- TreeSet de users ordenados pelo número de posts.

[1] Apenas utilizado se for resposta
[2] Apenas utilizado se for resposta
[3] Apenas utilizado se for pergunta
[4] Apenas utilizado se for pergunta

3.2.1 Decisões tomadas

Para facilitar o processo de alterar informação dos posts ou utilizadores, se tal for necessário no futuro, verificamo-nos de que a informação esteja sempre contida dentro da estrutura principal de dados, e que seja replicada o mínimo número de vezes.

Para manter a simplicidade e elegancia do código, decidimos que seria melhor não fazer uso de TreeMaps, já que as estruturas são construidas incrementalmente, pelo que seria necessário atualizar a ordem do Map quando se adicionava informação à base de dados.

3.3 Estratégias seguidas em cada uma das interrogações

- Interrogação 1: acede ao post a partir da tabela de hash de posts, e extrái a informação pedida.
- Interrogação 2: retorna a sublista (N primeiros elementos) da lista de utilizadores ordenados pelo número de posts, contruida a partir de um TreeSet na estrutura principal.
- Interrogação 3: itera pelos posts entre as duas datas (guardados num TreeSet para uma extração eficiente da lista), e mantem uma contagem das perguntas e outra das respostas.
- Interrogação 4: itera pelos posts entre as duas datas, filtra aqueles que não são perguntas ou não contém a tag especificada, e controi a lista dos restantes (ainda ordenados).
- Interrogação 5: acede ao utilizador com o id especificado para extrair a short bio, e controi a lista dos últimos 10 posts a partir do TreeSet dos posts do utilizador ordenados cronologicamente.
- Interrogação 6: itera pelos posts entre as duas datas, adicionando cada um a um conjunto (TreeSet) ordenando-os pela sua pontuação.
- Interrogação 7: itera pelos posts entre as duas datas, filtrando os posts que não são perguntas, e adicionando cada um a um conjunto (TreeSet) ordenando-os pelo seu número de respostas.
- Interrogação 8: itera por todos os posts (ordenados por data), filtra os que não são perguntas e as perguntas que não contêm a palavra especificada no título. Retorna os N mais recentes.
- Interrogação 9: acede à lista de posts (ordenados por data) de cada um dos utilizadores especificados e converte cada um no id da pergunta associada (o próprio id caso seja pergunta, ou o id da sua pergunta caso seja resposta). Depois são retirados de uma lista os elementos que não pertencem à outra, e por fim são retirados os id duplicados, mantendo sempre a ordem cronológica.
- Interrogação 10: acede ao post especificado pelo seu id, e itera pelas suas respostas (guardadas num conjunto), determinando qual delas tem a maior pontuação segundo a fórmula da interrogação.
- Interrogação 11: constrói a lista dos N utilizadores com maior reputação a partir de um conjunto ordenado construido anteriormente. itera pelos posts entre as duas datas, e filtra aqueles em que nenhum dos N utilizadores participou. Mantem um HashMap de ids de tags para suas contagens. Depois de contar todas as ocorrência das tags na lista de

posts filtrados, adiciona cada tag no HashMap a um conjunto ordenado com base na sua contagem. Por fim, é retornada a lista das N tags mais utilizadas desse conjunto, ainda ordenadas.

3.4 Estratégias para melhoramento de desempenho

Em relação à primeira fase, o desempenho das queries em si melhorou bastante, e o tempo de inicialização não sofreu penalização devido ao uso do método de parsing SAX, que reduz a quantidade de memória utilizada e é mais eficiente.

Para conseguir isto, em vez de, por exemplo, ordenar os posts por data cada vez que se faz uma query, os posts são introduzidos na estrutura de dados e ordenados nesse momento. Esta antecipação permite não duplicar o esforço nas queries, não só para ordenar posts por data, mas também para os ordenar por número de respostas e para ordenar os utilizadores por número de posts e por reputação.

Capítulo 4

Conclusão

Neste projeto foi-nos pedido responder a um conjunto de interrogações acerca de comunidades do Stack Exchange a partir dos dados disponibilizdos em formato XML. Na primeira fase utilizamos a linguagem de programação *C* com recurso a várias bibliotecas de código, e na segunda *Java* e a sua biblioteca standard.

A primeira fase do projeto foi a que demorou mais, tanto pela dificuldade de resolver o problema proposto como pela natureza da linguagem utilizada. A segunda fase foi mais simples pois já tinhamos a ideia do que fazer, apenas faltou melhorar a implementação, e a linguagem utilizada é manifestamente mais acessível. Na segunda fase, seria possível tornar o código mais idiomático, utilizando as exceções do *Java*, e talvez criando subclasses distintas para perguntas e respostas.