Universidade do Minho

Mestrado Integrado em Engenharia Informática

Projeto com objetivo de gerir Snapshots Wikipedia

Laboratórios de Informática III

Grupo 111

Trabalho realizado por:

Filipe Fortunato A75008
João Leal A75569
Manuel Monteiro A74036

1 de Maio de 2017

Conteúdo

Capítulo 1

Introdução

No âmbito da unidade curricular de Laboratórios de Informática III foi proposto um projeto para filtrar uma grande quantidade de dados provenientes das *Wikipedia*, que tem como objetivo enriquecer os conhecimentos adquiridos nas UCs de Programação Imperativa, de Algoritmos e Complexidade. Um dos principais desafios deste projeto é a programação em grande escala pelo facto de termos de filtrar uma grande quantidade de dados.

Capítulo 2

Abordagem do Problema

Após uma análise do enunciado, podemos perceber que este trabalho poderia "dividir-se"em duas partes, a primeira em que através do *parser* filtramos os ficheiros em XML dados pelos professor e criamos algumas estruturas para podermos guardar a informação necessária, e em segundo lugar trabalhar essa informação através das interrogações feitas pelo professor. Abaixo encontram-se todos os passos que o nosso grupo tomou para a resposta a todas as interrogações.

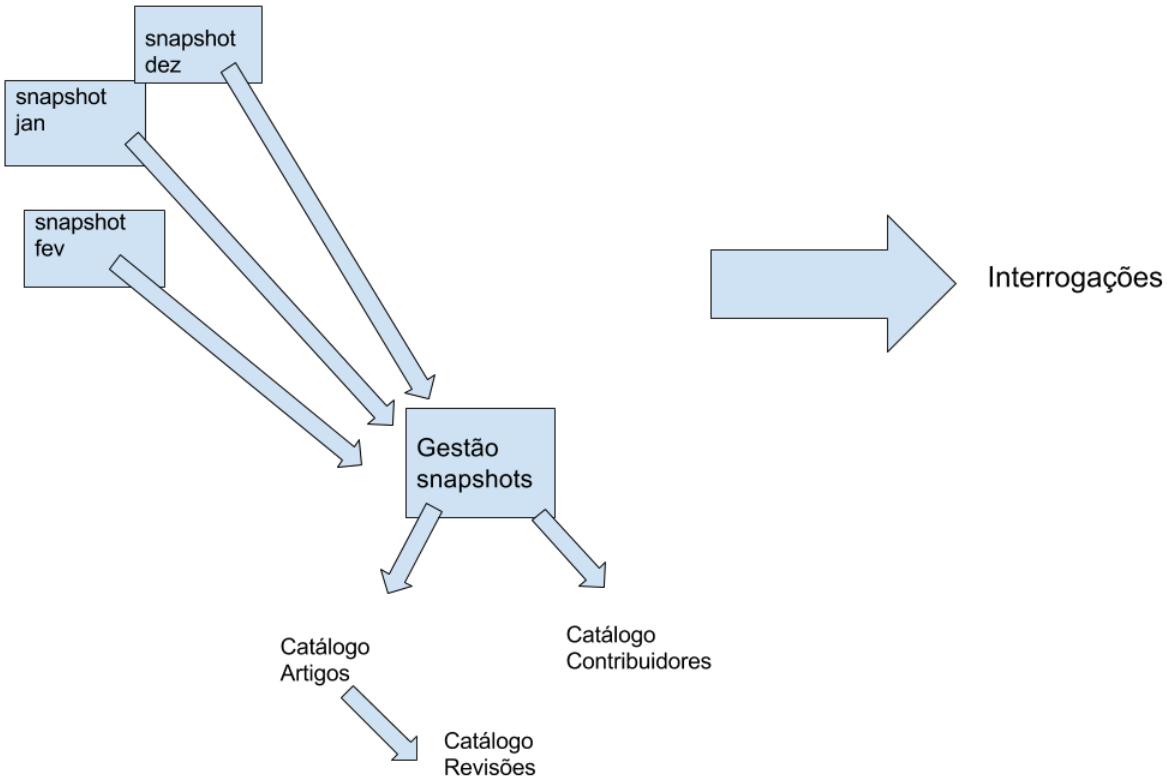

Figura 2.1: Estrutura do trabalho

2.1 Parser

Como os snapshots fornecidos pelo professor se encontravam em XML, usamos a libraria *Libxml2* para podermos fazer para podermos filtrar os dados para de seguida os inserirmos, artigo a artigo, nas nossas estruturas.

2.2 AVL

Para podermos guardar a informação fornecida pelo *parser* criamos uma Árvore Binária Balanceada. Esta AVL está organizada pelo ***long* valor**, contém também ***void** **info** um apontador para uma estrutura, que no caso dos artigos contém a informação dos artigos, no caso dos contribuidores contem a informação acerca destes e para as revisões acontece o mesmo. Para além disso contem um ***int* height** que é usado no balanceamento da AVL e por fim contem também **struct nodo *left, *right** que indica o resto da AVL.

```
struct nodo{
   long valor;
   void *info;
   int height;
   struct nodo *left,*right;
};

struct avl{
    int size;
    int duplicates;
    NODO head;
};
```

Figura 2.2: Estrutura da AVL

2.3 Catálogo de Artigos

Esta estrutura é um Array de AVL's que está organizada pelo primeiro digito de cada id correspondente ao artigo. Nos nodos de cada AVL tem um ***char** **title** que indica o titulo do artigo, ***long* max b** que indica o tamanho da revisão de um artigo em bytes, ***long* max w** que indica o numero de palavras que de cada revisão, e também **CAT R revisoes** que contem as revisões de cada artigo. Para além disso o Catálogo de Artigos ainda contem ***int* all r** que indica o numero total de revisões.

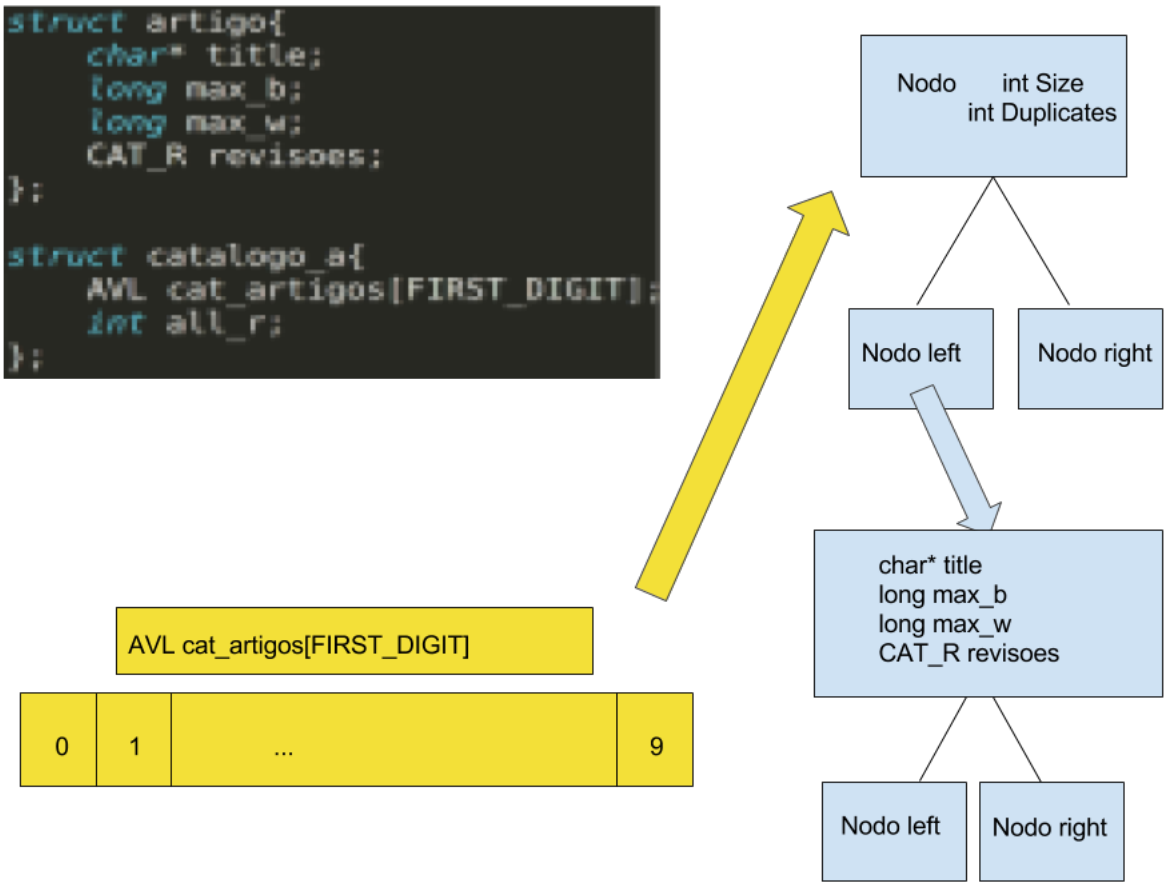

Figura 2.3: Estrutura da AVL

2.4 Catálogo de Revisões

Esta estrutura engloba a informação correspondente as Revisões de cada artigo e está organizada pelo id do artigo. Cada nodo da AVL vai conter ***char**** **timestamp** que indica a data em que a revisão foi feita.

```
struct revisao{
    char* timestamp;
};
```

Figura 2.4: Estrutura da AVL de Revisões

2.5 Catálogo Contribuidores

Esta estrutura é um array de AVL's que guarda toda a informação acerca dos contribuidores. Cada AVL está organizada pelo id do contribuidor. Cada nodo da AVL contém ***char**** **username** que indica o nome do contribuidor e também o ***int*** **total revisions**.

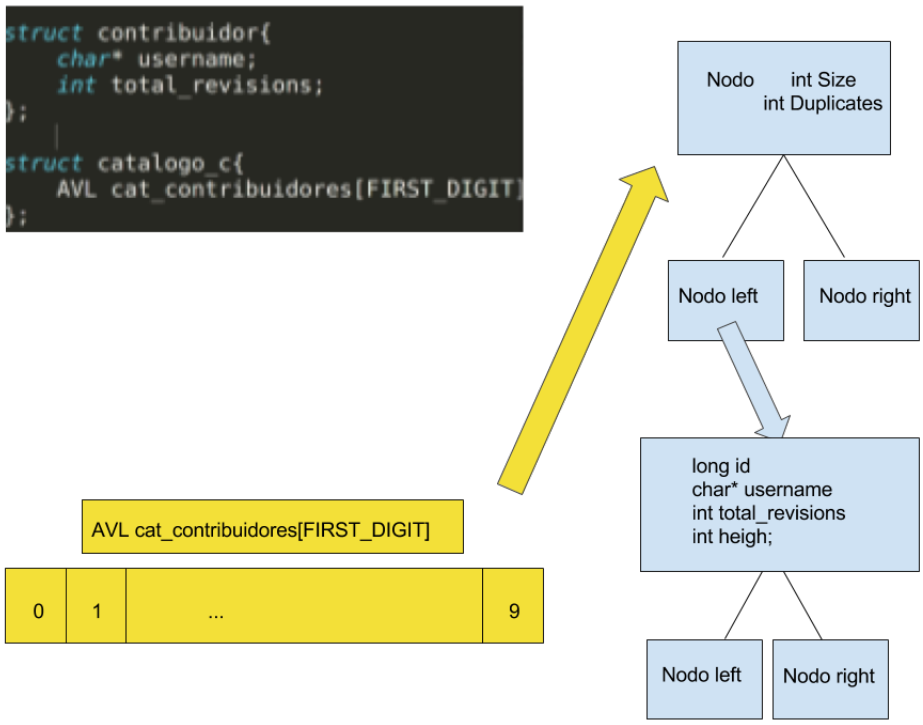

Figura 2.5: Estrutura do catálogo de Contribuidores

2.6 Utility

No ficheiro utility.c encontram-se algumas funções usadas nas estruturas faladas acima.

Capítulo 3

Interrogações

3.1 Interrogação Número 1

Esta interrogação pede para retornar todos os artigos encontrados nos backups analisados. São também contabilizados todos os artigos duplicados.

Estratégia: Ao criarmos a nossa estrutura *Catalogo Artigos* incluímos um *int* all r que é incrementado ao inserir um artigo na estrutura.

3.2 Interrogação Número 2

Esta interrogação procura saber o número de artigos distintos presentes em todos os snapshots analisados.

Estratégia: contar o número de elementos presentes na estrutura *Catalago Artigos*, pois na nossa estrutura não estão incluídos os artigos duplicados.

3.3 Interrogação Número 3

Esta interrogação procura contar as revisões presentes nos snapshots analisados, retornando apenas o número de revisões únicas.

Estratégia: Na nossa estrutura AVL temos um *int duplicates* que é incrementado sempre que exista uma revisão duplicada. Para a resolução desta interrogação basta nos portanto subtrair o numero total de revisões (*all r*) pelo número de revisões duplicadas (*int duplicates*)

3.4 Interrogação Número 5

Esta interrogação devolve o nome de um contribuidor correspondente a um dado identificador de contribuidor, sendo que caso não haja um contribuidor com o identificador explicitado, é retornado o valor NULL.

Estratégia: Para a resolução desta interrogação, é feita uma procura pelo id do contribuidor pela estrutura do *Catalogo Contribuidores*, pela função *getContribuidor*. Depois de termos o contribuidor, basta apenas, através da função *getUsername* encontrar o nome do contribuidor com um determinado id.

3.5 Interrogação Número 7

Esta interrogação devolve o título de um artigo correspondente ao id desse artigo.Caso não haja um artigo com o identificador explicitado, é retornado o valor NULL e caso um artigo tenha várias revisões, é considerado o título da revisão mais recente do artigo.

Estratégia: Para a resolução desta interrogação, é feita uma procura pelo id do artigo pela estrutura do *Catalogo Artigos*, pela função *getArtigo*. Depois de termos o artigo, basta apenas, através da função *getTitle* encontrar o titulo do artigo com esse determinado id.

3.6 Interrogação Número 10

Esta interrogação devolve o timestamp para uma certa revisão de um dado artigo, sendo que caso não haja a revisão para o artigo específico, é retornado o valor NULL.

Estratégia: Dado o id do artigo e um id de revisão, procuramos na estrutura *Catalogo Artigos* a existência do artigo com o identificador dado. Se este não existir é retornado o valor NULL. Se o artigo estiver presente na estrutura retiramos as revisões correspondentes ao artigo com o id dado *getArtigoRevisoes*. Posteriormente procuramos pelo id nas revisões desse artigo *getRevisao*. Finalmente é so retirar o timestamp dessa revisão *getTimeStamp*.

Capítulo 4

Conclusão

As nossas maiores dificuldades na realização deste projeto foram a de estruturar o trabalho de maneira a que a este fosse eficiente, pois o elevado volume de dados exige a que tenhamos muito cuidado com a estruturação do nosso código para permitir uma maior rapidez no programa e foi isso que nos dificultou há realização das interrogações.