Universidade do Minho

Mestrado Integrado de Engenharia Informática

Laboratório de Informática III

PROJETO PRÁTICO JAVA

(MIEI – LI3 20/21)

Grupo Nº40:

- Pedro Aquino Martins de Araújo - A90614

- Rafael dos Anjos Arêas - A86817

- Vitor Lelis Noronha Leite - A90707

Data de Entrega: 19/06/2021

ÍNDICE

- Visão geral do projeto e MVC……...2
- Descrição dos módulos……..3
- Descrição das querys…….4
- Estatística……..4
- Testes…….6
- Conclusões e desafios……....8

Visão geral do Projeto e MVC

Tendo em conta o funcionamento do projeto, de início temos o carregamento de dados, onde o usuário possui duas opções: carregar a partir de um ficheiro binário (** opção que está a dar erros de compilação) ou carregar a partir dos ficheiros csv , e sempre que envolve o path do arquivo, é dado como opção o próprio usuário passar o caminho ou assumir o padrão. De seguida após o carregamento, o usuário possui 4 opções de funcionalidades: Carregar novamente os dados através de um ficheiro binário(problemas de heap space) , Gerar um ficheiro binário a partir dos dados que já foram carregados, Ver as Estatísticas, Ver as querys disponíveis.

O projeto foi desenvolvido a partir de um modelo MVC, onde o controller é a ligação entre o model e o view , de forma que os requisitos que a view necessita do model são atendidos pelo controller, que irá pedir ao model e irá entregar à view. O controller foi tratado como o módulo principal do projeto, onde executa as principais tarefas. Optamos também por colocar o módulo de carregamento de dados do ficheiro na pasta controller, por considerarmos o carregamento como fonte de recursos do controller.

Estruturas de dados utilizadas

Utilizamos o Map como principal estrutura para carregamento e iteração de dados, operamos com maior frequência o HashMap , mas também utilizamos o TreeMap quando era preciso de ordem.

O Motivo de escolha do Map , foi devido à facilidade de iteração entre os elementos com ajuda de seus métodos disponíveis, além de um ótimo tempo de execução constante para pesquisas de valores.

Descrição dos Módulos

GestReviewApp:

- Módulo principal da execução. Atua como a "Main" do projeto.

Testes:

- Usado para testes de performance do código.

View:

- **I0 :** possui toda parte de iteração com o usuário

Model:

- **GestReviewModel:** módulo principal que agrega todas a estruturas do model
- **Estatística:** contém toda estrutura e métodos relacionados à Estatística
- **Users:** agrega todas as funcionalidades relacionadas aos dados lidos do ficheiro dos usuários. Sejam elas da classe User (que lidam com um único usuário) ou da MapUsers e sua Interface (que lidam com os vários usuários no ficheiro)
- **Business:** parecido com o módulo Users, este módulo também possui todas as estruturas de dados só que relacionado aos negócios desta vez. Sendo elas em sua classe Business ou MapBusiness com sua interface que executam funções parecidas com as citadas no módulo Users
- **Review:** contém todo código e funcionalidades entre os dados das avaliações e que de forma parecida com os dois módulos anteriores podem ser encontrados na classe Review ou MapReview e suas respectivas interfaces

Controller:

- **MainController:** agrega os principais componentes de model e view, e faz a execução do projeto.
- **DataLoader:** faz todo o carregamento de dados a partir da leitura dos ficheiros passados pelo usuário.

Descrição das querys

Query 1:

- **Estratégia adotada:** É reutilizado o método bus_non_available de Estatística, que gera a lista ordenada de ids de negócios nunca avaliados. Feito a partir de streams() iterando sobre os maps de review e business.

- **Comentários:** Lentidão para carregamento de dados de ficheiros grandes, principalmente do catálogo users, o que já era considerado esperado dado o tamanho deste. Tempo de execução: 20 segundos.

Query 2:

- **Estratégia adotada:** Utilizou-se um iterator que percorre os reviews e povoa uma lista de Strings contendo todos os users ids , que pertencem à determinado ano e mês. E depois é feito a partir de stream() um filtro dos diferentes ids.

Estatística

Grupo 1:

- **Nome de cada ficheiro:** Foi utilizado uma lista de strings , carregado na leitura dos ficheiros.
- **Nº total de registros errados do review:** Foi utilizado um int que é incrementado na verificação dos registros do review na leitura dos ficheiros.
- **Nº total de negócios**: Foi utilizado um int que é incrementado a cada registo passado na verificação na leitura dos ficheiros.
- **Nº total de negócios diferentes que foram avaliados:** Calculado no método bus_non_available , que antes de retornar a lista dos que não foram avaliados , é gerado a lista total dos avaliados distintos, utilizando stream().
- **Nº total de negócios não avaliados:** É calculado após chamado o método bus_non_available que retorna a lista de todos os ids de negócios não avaliados, é feito apenas um size().
- **Nº total de users:** Foi utilizado um int que é incrementado a cada registo passado na verificação na leitura dos ficheiros.
- **Total de users inativos (sem reviews):** Calculado no método activeUsers, onde é realizado um stream() no Map contendo os reviews.
- **Total de reviews com 0 impacto:** É utilizado um contador na na verificação de registos na leitura dos ficheiros, onde é incrementado caso a soma de funny cool e useful for == 0.

Grupo 2:

Questão A: Utilizou-se o TreeMap "reviewMes" que usa como chave o mês de quando o review foi feito para facilitar os cálculos de cada mês, sendo populado com dados no carregamento do ficheiro, optimizando assim o número de travessias. A partir do método "totalReviewMes", cria-se um outro TreeMap ainda com o mês sendo a chave e o valor sendo o total de reviews feitos naquele mês. Isto é feito através de uma aplicação do método ".forEach()" no "reviewMes" que para cada chave nele (o mês) utiliza-se o método ".put()" no novo Map, onde os parâmetros seriam a própria chave a quantidade de reviews que lá tinham calculados pelo método ".size()".

Questão B: Nesta etapa utilizou-se uma estratégia parecida com a anterior para calcular média mensal, criou-se também um TreeMap com os meses como chave mas os valores passaram a ser a soma das Stars dos reviews daquele mês calculada através do método ".reduce()". Além disso, a variável "mediaRevGlobal", através do mesmo método, percorre o novo Map e soma todas os valores, tendo assim uma soma total.

Contudo, o que fora encontrado é apenas a soma dos valores e não a média como foi pedido. Logo, para calcular a média, recorre-se novamente para o "reviewMes", onde dividimos a "mediaRevGlobal" pelo número total de reviews (encontrando a média) e no Map com as médias mensais aplicamos o método ".replace()" que substituirá o atual valor daquela chave pela sua divisão com o número de reviews daquele mês.

Testes

Foram realizados testes de performance nas principais partes do projeto. Destaque para os testes feitos no carregamento de dados dos ficheiros, onde foi realizado diferentes estratégias e estudos para obter uma melhor performance. Segue os resultados de cada parte:

Carregamento dos ficheiros:

- **Scanner(FileInputStream) :**
  Tempos médios de leitura de cada ficheiro:
  business_full.csv: 1 sec
  reviews_1M.csv: 27 sec
  users_full.csv: 5 minutos

  Observações: Apesar de consumir pouca memória (devido a leitura de cada linha sem guardar uma referência e apontador a elas, ou seja, sem guardar em memória) , é muito lento a execução de leitura.

- **ReadAllLines (File) :**
  Tempos médios de leitura de cada ficheiro:
  reviews_1M.csv: 13 sec

  Observações: Apesar de ser muito rápido a leitura, consome bastante memória (devido a leitura total das linhas, podendo haver algum colapso de memória, dependendo da dimensão do arquivo). Não foram realizados testes nos outros ficheiros, devido à preocupação em relação à memória, e principalmente porque foi encontrada uma solução considerada ótima que vem a seguir.

- **BufferedReader : ****
  Tempos médios de leitura de cada ficheiro:
  business_full.csv: 0.4 sec
  reviews_1M.csv: 15 sec
  users_full.csv: 41 sec

  Observações: Faz uma leitura eficiente das linhas e caracteres a partir de buffers, atendendo à uma boa performance em relação ao tempo de execução e de consumo de memória.

Leitura e escrita de dados em ficheiro auxiliar :

Foi possível obter resultados apenas da escrita, pois a leitura não conseguimos executar , devido a erros de compilação e problemas com heap space.

- **Tempo médio de escrita utilizando ObjectStream:** 3 min
- **Observações:** Consideramos muito lento a execução, acreditamos que deve haver alternativas melhores, porém, buscamos mas não conseguimos obter êxito na procura.

Estatísticas:

Grupo 1 :

- **Tempo médio de execução:** 3 sec

Grupo 2 :

- **Tempo médio de execução:** 0.5 sec
- **Observações:** Obteve-se um ótimo tempo de execução , devido a optimização que foi realizada, onde foi criado um map na leitura dos ficheiros, que separa os reviews por mês. Obtendo-se então , um tempo de execução constante .

Queries:

1. **:**
   - **Tempo médio de execução:** 3 sec

2. **:**
   - **Tempo médio de execução:** 0.2 sec

Conclusões e desafios

Fazendo uma análise geral dos objetivos que tínhamos à ser cumpridos, conseguimos concluir: validação e carregamento de dados dos ficheiros, escrita de dados em um ficheiro binário, grupo 1 das estatísticas completo , grupo 2 das estatísticas ficou por fazer apenas calcular o total de usuários diferentes que avaliaram por mês, das querys concluímos a 1 e 2.

Este projeto nos ajudou a perceber não só a gestão e tratamento de dados através da linguagem Java, mas também a compreender e aplicar melhor conceitos como o MVC, os métodos aplicados em Maps e a leitura de arquivos com formas distintas.

Contudo, assim como no primeiro projeto prático, nosso maior desafio foi novamente a gestão de tempo por nossa parte que nos limitou a um número menor de querys concluídas e a falta do último quesito do grupo 2 das estáticas como dito no primeiro parágrafo.

Concluímos então que apesar dos desafios mencionados conseguimos realizar o que nos propusemos a fazer e conseguimos assim, adquirir mais conhecimento sobre.