Desafio FullStack - The Hitchhikers Guide to Facts and Schema

1. Instruções de uso

A solução encontra-se hospedada na plataforma em nuvem Heroku. Para acessá-la, basta utilizar o link: https://facts-and-schema.herokuapp.com/. No caso de não funcionamento da plataforma por motivos adversos, uma solução é utilizar a versão do código que encontra-se no Github: https://github.com/rodolphorosa/desafio-intelie. Nesse caso, após clonar o código, é necessário abrir o terminal, direcioná-lo para a pasta raiz do código, executar o comando `python app.py` e acessar o endereço: localhost:5000.

Na ausência das bibliotecas Flask, lxml e MarkupSafe, será necessária a criação de um ambiente virtual para que a solução possa ser executada em sua máquina. Nesse caso, deve-se executar os seguintes comandos no Linux (no diretório do projeto):

```
$ virtualenv -p python3 envname
$ source envname/bin/activate
$ pip install -r requirements.txt
$ python app.py
```

Para testar apenas a funcionalidade destinada à resolução do problema de encontrar fatos vigentes,, é necessário unicamente executar o módulo *schemaFacts.py* utilizando o comando `python schemaFacts.py` na linha do comando do Windows/Linux.

Ao abrir a página web da solução, será necessário realizar *log in*. Há dois usuários registrados: user e visitor, ambos com senha password. O primeiro tem como *role* “admin”, enquanto o segundo, “visitor”. Dessa forma, *user* terá acesso a todas as funcionalidades do sistema, ao passo que *visitor* terá acesso apenas às funcionalidades de visualização. Após efetuar login, o usuário é direcionado para a página principal da aplicação, que apresenta a lista de todos os fatos existentes. Na parte superior, estão disponíveis as opções *Log out*, *Home*, *See Schema*, *See Current Facts*, nessa ordem. Ao acessar See Schema, o usuário terá acesso à lista de atributos dos dados e as opções de adição, atualização e deleção de atributos. De forma análoga, ao acessar, See Current Facts, abrir-se-á uma página contendo todos os fatos vigentes; ao usuário serão oferecidas as opções de adicionar e deletar um fato. Ao clicar no nome de qualquer uma das entidades listadas, o usuário será direcionado para uma página onde é mostrado o histórico de manipulações realizada

naquela entidades, isto é, as adições de novos atributos e deleção de um fato a ela associado. Ao realizar *log out*, o usuário é direcionado imediatamente para a tela de *log in*.

2. Descrição da solução

Para a resolução do problema proposto, foi desenvolvido o módulo *schemaFacts*. Esse módulo possui uma classe nomeada *SchemaFacts* cujo construtor tem como parâmetros 1) uma lista de tuplas de atributos (*schema*) e 2) uma lista de tupla de fatos (*facts*). Seu principal método, intitulada *get_current_facts*, é responsável por retornar os fatos vigentes segundo as especificações do desafio. O pseudocódigo deste método encontra-se é descrito a seguir:

```
1. Crie uma lista dos fatos com valor added igual a True
2. Crie uma lista dos fatos com valor added igual a False

3. Inicie uma lista vazia de fatos vigentes
4. Para cada atributo do esquema:
5.     selecione todos os fatos deste atributo da lista de fatos com added True
6.     se o atributo tiver cardinalidade igual a 1:
7.           para cada entidade na lista de fatos desse atributo:
8.                 adicione o último fato desta à lista de fatos vigentes
9.     senão
10.          adicione todos os fatos deste atributo à lista de fatos vigentes

11. Para cada fato com added False
12.    para cada fato da lista de fatos vigentes:
13.          se os dois fatos compartilharem a mesma entidade, atributo e valor:
14.                remova o fato da lista de fatos vigentes

15. Retorne os fatos vigentes
```

Sejam N e M as entradas do método, isto é, as listas de fatos e de atributos (esquema), respectivamente. Seu tempo de execução é descrito como segue:

$$f(m, n) = 2n + m(x + z) + yw,$$

sendo “x” e “y” o tamanho das listas de fatos com campo added igual a True e False, respectivamente, “z” o número de entidades e “w” o número de fatos vigentes, cuja magnitude dependerá da natureza dos dados. Desse modo, a complexidade assintótica desse método é **O(n + m)**.

Além desse método, a classe *SchemaFacts* possui uma quantidade considerável de outros métodos que auxiliam na resolução do principal e para manipulação desses dados, os quais fizeram-se necessários para o desenvolvimento das demais rodadas do desafio. A complexidade desses métodos encontra-se listada a seguir:

Método	**Complexidade**, sendo *n* o número de fatos e *m*, o número de atributos do esquema.
__retrieve_facts_by_entity	O(n), pois é necessário ler toda a lista de fatos.
__retrieve_facts_by_attribute	
__retrieve_deleted_facts	
__retrieve_non_deleted_facts	
__retrieve_attribute_by_name	O(m), pois é necessário ler, no pior caso, todo o esquema.
__drop_facts_by_attribute	O(n), pois é necessário ler todos os fatos.
__drop_facts_by_entity	
get_schema	O(1)
get_facts	O(1)
get_attribute	O(m), similar a *__retrieve_attribute_by_name*
insert_attribute	O(m), pois é necessário verificar se atributo já existe.
insert_fact	O(m), pois é necessário varrer a lista de

update_attribute	atributos
delete_attribute	
delete_fact	O(1)

3. Pontos Fortes e Fracos e Futuras Melhorias

Dentre os principais pontos fortes da solução encontram-se: a sua eficácia e escalabilidade, haja vista que as operações utilizadas são de baixa complexidade e, juntas, foram eficazes em retornar os resultados esperados; a qualidade do código, o qual utilizou orientação a objetos e não somente programação estruturada, o que facilita a sua manutenção e desenvolvimento de novas funcionalidades; e a lógica empregada, que fez uso de mecanismos simples, como *filter*, e a maneira como todas as informações que compõem os dados foram utilizadas em conjunto. Como ponto fraco, pode ser citado a rigidez da solução, que foi projetada para um tipo de específico de dado, o que faz necessária uma reestruturação caso o modelo de dados mude. Além disso, a solução ainda possui limitações quanto ao paralelismo das operações que poderão ser feitas sobre os dados em um ambiente de produção.

Como futura melhoria, pode-se considerar a utilização de programação concorrente para lidar com as inúmeras adições, atualizações e deleções dos atributos do esquema e manipulações nas entidades realizadas em paralelo. Apesar da metodologia utilizada ter se demonstrado eficaz, outra possível melhoria seria em eliminar o número grande de varreduras feitas sobre os dados, o que tornaria o processo mais rápido.