Projeto Final - MC536 Grupo 9

Descrição do problema e soluções encontradas

É difícil seguir uma dieta balanceada com todos os nutrientes necessários e relacionar isso com os alimentos que contêm tais nutrientes, para sua inclusão nas refeições diárias. Além disso, seguir uma alimentação com tudo que é necessário pode se tornar cara. Uma das soluções atuais é a utilização de complexos vitamínicos para suprir as necessidades que uma pessoa precisa. Entretanto, é um suplemento alimentar que não custa barato, além de ser artificial.

O projeto consiste em encontrar uma complementação alimentar utilizando apenas frutas e sucos para suprir as necessidades diárias de algumas vitaminas e alguns minerais, de acordo com faixas etárias pré definidas. Além de ser acessível a um maior grupo de pessoas, é uma solução que pode ser usada não apenas por pessoas interessadas em melhorar a alimentação, mas também por clínicas especializadas na saúde alimentar.

Extraímos, para tanto, dados referentes à necessidade diária de vitaminas e minerais, por faixa etária, contendo as quantidades ideal e máxima para cada nutriente, e dados nutricionais de alimentos, especificamente frutas e sucos, para 100 gramas de cada alimento. Juntando essas informações, geramos uma lista com os alimentos que suprem as necessidades nutricionais diárias de uma pessoa de acordo com sua faixa etária, cobrindo a quantidade ideal estipulada, mas sem nos preocuparmos com a quantidade máxima pois, como só extraímos frutas e sucos, as combinações de alimentos eram limitadas. Além disso, a quantidade em massa da lista total de alimentos foi superior a 2kg, também por termos escolhido somente frutas e sucos, que não são suficientemente capazes de suprir as necessidades nutricionais de forma eficiente. Por exemplo, para um *teenager* (idade entre 14 e 18 anos), a quantidade indicada é de 75 mg de vitamina C e uma quantidade máxima de 1800 mg diariamente. A acerola possui 1677 mg de vitamina C em 100 gramas, estando no intervalo maior que a quantidade ideal, sendo um alimento recomendado.

Mas podem surgir alimentos que não são comuns de serem encontrados, ou alimentos que são inviáveis para uma pessoa, ou até mesmo empecilhos relacionados a gostos pessoais.

Assim, a aplicação também é para esse caso e é possível encontrar outras relações, como alimentos que são nutricionalmente equivalente em relação a um nutriente específico. Portanto, há uma flexibilidade na escolha dos alimentos. Essa equivalência foi definida como sendo uma faixa aceitável de quantidade de um certo nutriente. Esta faixa compreende uma porcentagem definida para mais ou para menos, por exemplo, para 10%, morango que contem 31,7 mg de vitamina C e limão com 29,1 mg são equivalentes em relação ao suprimento de vitamina C.

Com as nossas bases de dados de alimentos e nutrientes e suas relações, assim como as relações de equivalência, o usuário é capaz de procurar alimentos que satisfaçam certas necessidades nutricionais e criar uma lista de alimentos que possam complementar de maneira natural e eficiente tais necessidades e quando certos tipos de alimentos não forem viáveis por diferentes motivos como preço, disponibilidade, o grafo permite que novos alimentos sejam encontrados mantendo a qualidade da lista construída primeiramente

Arquitetura do sistema

Para mostrar de uma forma resumida o visual do projeto, o diagrama abaixo traz os problemas apresentados e como foram solucionados com os arquivos criados.

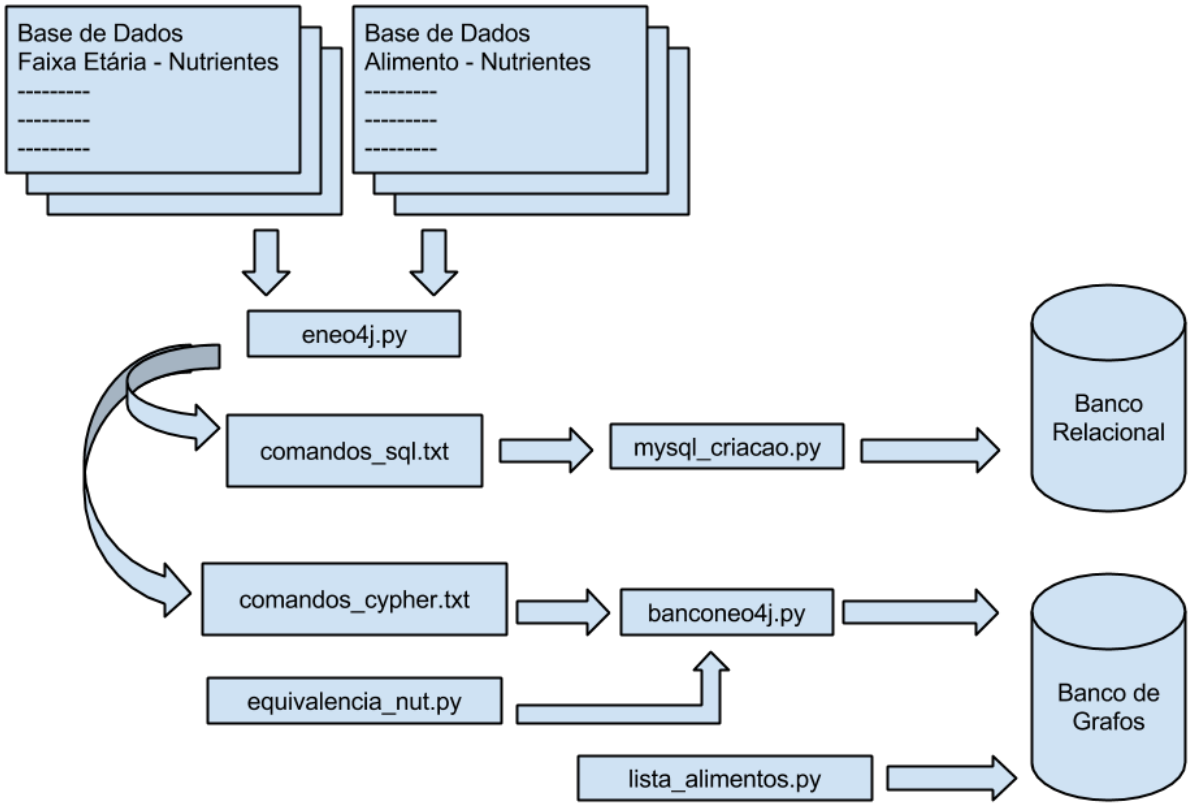

Figura 1: diagrama do projeto.

Modelo Entidade - Relacionamento e Relacional

-Modelo relacional:

FaixaEtaria(id_idade)

Alimento(id_alimento)

Alimento_Nutriente(id_alimento, id_nutriente, quantidade)

Nutriente(id_nutriente)

FaixaEtaria_Nutriente(id_nutriente, id_idade, quantidade_ideal, quantidade_maxima)

Modelo ER:

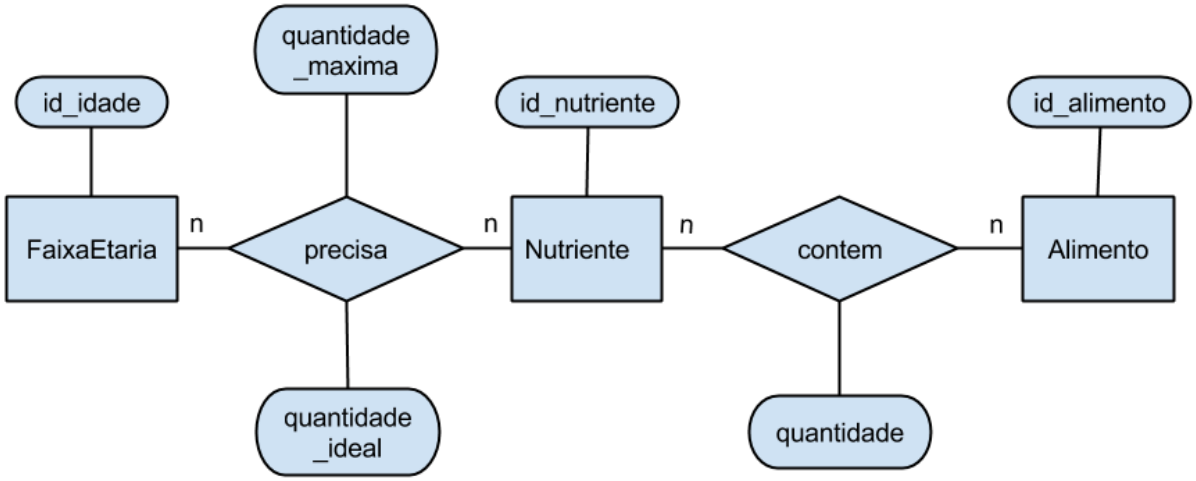

Figura 2 - Modelo ER.

Desafios da extração e *data cleaning*

Para implementar todo o sistema, foi investido mais tempo nas informações relevantes que eram extraídas. As bases de dados não estavam formatadas do jeito esperado, assim, era recorrente encontrar erros nos dados ou dados inconsistentes.

Algumas *tags* utilizadas nos *sites* não eram claras, algumas estavam com sintaxe errada, assim, sendo necessário alteração na lógica do código para extração dos dados. Para apresentação dos dados no *site*, muitas vezes eram utilizados caracteres de espaço e tabulação para deixar as informações alinhadas nas tabelas, que dificultavam a extração.

O maior problema encontrado pelo grupo, foi achar uma unidade geral para os nutrientes. Para apresentar de uma forma mais fácil para leitura das tabelas nos *sites*, as unidades variavam entre micrograma, miligrama, grama, IU (International Units). Porém, isso atrapalha as relações feitas entre as bases, pois não era possível relacionar as necessidades de um nutriente em uma unidade com as quantidades de nutrientes em um alimento em outra unidade. Assim, a solução mais apropriada era converter todas as unidades para micrograma, pois valores muito pequenos em outras unidades seriam desprezíveis.

Detalhamento do código

A extração do primeiro site foi aquela relativa às informações da tabela de nutrientes necessários por faixa etária. Para isto, usamos a ferramenta de Python BeautifulSoup. Primeiramente, determinamos as localizações das tags em html das informações na tabela que precisávamos para, a partir disso, selecionar as informações relevantes, isto é, a quantidade de necessária de cada nutriente para cada faixa etária. Porém, como foi dito anteriormente, a tabela no site estava bem mal organizada: a identação era feita de maneira deselegante, com diferentes quantidades de espaços em posições aleatórias, e as unidades muitas vezes tinham grandezas diferentes. Para isso, tivemos que utilizar bastante expressões em regex para uniformizar o formato da informação. Este foi um dos maiores desafios. Feito isso, o próximo passo era transformar as diversas unidades em uma unidade padrão, que escolhemos como sendo o micrograma.

A segunda base de dados foi aquela relativa aos alimentos e aos seus nutrientes. Ela possuía uma variedade enorme de alimentos, de modo que optamos por utilizar somente frutas e sucos, a fim de tornar a análise de dados mais simples e visualizável. Para isto, percorremos todas as páginas de alimentos, selecionando apenas um tipo de cada fruta, e ignorando alguns detalhes demasiadamente específicos delas. Por exemplo, no caso de *"Apples, raw, without skin, cooked, microwave",* ignoramos tudo depois da primeira vírgula, ficando somente com *"Apple".* Feito isso, o algoritmo entrava no link referenciado pelo alimento para entrar na página referente às informações nutricionais do alimento. Ao extrair os nutrientes, tivemos que comparar com aqueles extraidos da primeira base para que os dados ficassem consistentes, e para que a análise posterior pudesse ser feita.

Em relação à criação do banco de dados relacional foi escolhido o projeto MySQL para armazenar os dados coletados. A partir dos comandos gerados pela extração e armazenados em um arquivo de texto, que é passado como parâmetro logo na chamada desse *script*, foi utilizada a biblioteca MySQLdb que executa comandos gerados de *scripts* python. Com isso, são criadas todas as tabelas necessárias para inserção dos dados criados e a própria inserção dos dados.

Quando foi pensado em realizar a análise dos dados coletados e inseridos no banco de dados relacional, foi decidido o uso de grafo. Neste ponto, era desconhecido como seria feita a criação do banco de dados a partir do que havia sido feito até então. Foi necessário primeiramente entender como seria feita a transação do modelo, até o momento relacional, para um modelo de grafos, que ferramenta seriam usadas para criar tal banco e que bibliotecas seriam usadas para criar *softwares* de aplicação que interajam com os dados.

O projeto Neo4j foi escolhido para criar o banco e assim foi preciso o aprendizado da linguagem Cypher para a criação do grafo e a ferramente Py2neo foi utilizada para fazer o interfaceamento dos dados no banco com uma aplicação em python.

No próprio site do Neo4j há certos estudos e explicações de como modelos relacionais são mapeados em modelos de grafos. O banco foi criado com sucesso e sem erros para, posteriormente, como já explicado, criar as relações de equivalência entre alimentos.

Assim, o banco de dados estava pronto para receber buscas de listas de alimentos que retornem resultados úteis e tangíveis.

O *script* python que cria as relações ‘equivalente’ recupera a lista de nutrientes do grafo e as listas de alimentos que contêm tais nutrientes e suas respectivas quantidades. As relações são baseadas em diferenças percentuais pré definidas e tendo os vetores das quantidades de nutrientes por alimentos ordenados é possível verificar se há essas equivalências sem ser necessário percorrer o vetor inteiro para cada nutriente.

A partir de consultas ao banco de dados, fizemos um algoritmo em Python que retorna uma lista com alimentos que supre as necessidades nutricionais diárias de cada faixa etária avaliada. Essa lista é feita assim: para cada nutriente, consultamos no banco qual alimento tem a maior quantidade daquele nutriente, criando um vetor com tais alimentos. Em seguida, fizemos consultas no banco para retornar a quantidade de nutrientes que cada um desses alimentos possuem. Então, para cada nutriente, calculamos as soma das quantidades de nutriente de cada alimento. Assim, comparando-se as somas para cada nutriente com suas quantidades ideais diárias necessárias, para cada faixa etária, pudemos identificar quais nutrientes foram supridos com a lista de alimentos encontradas e qual não foram. Para os que ainda não tinham sido supridos, fizemos uma consulta ao banco de dados de todos os alimentos que continham aqueles nutrientes em ordem decrescente de quantidade. Assim, selecionávamos os alimentos que continham a maior quantidade de um determinado nutriente, que ainda não estavam na lista, um por um; a cada alimento adicionado, verificávamos se a quantidade no nutriente em questão já havia sido suprida; se já, passávamos para o próximo nutriente. O trecho do código para o preenchimento da lista dos nutrientes não supridos, se encontra abaixo:

```
      for nut in range(len(nomesNutrientes)):
          somaNutAtual = somasNutrientes[nut]
          if somaNutAtual < qtidealNutrientes[nut]:
            query = 'MATCH (a:Alimento)-[r]->(n {id:\''+nomesNutrientes[nut]+'\'})
RETURN a.id, r.quantidade ORDER BY r.quantidade DESC'
            resultAlimentosFaltantes = graph_db.cypher.execute(str(query))
            for a in range(len(resultAlimentosFaltantes)):
             nomeAlimento = str(resultAlimentosFaltantes[a][0])
             if somaNutAtual >= qtidealNutrientes[nut]:
               break
             elif nomeAlimento not in alimentosIdeaisDistintos:
               alimentosIdeaisDistintos.append(nomeAlimento)
               query = 'MATCH (a:Alimento {id:\''+nomeAlimento+'\'})-[r]->(n) WHERE
n.id <> \'Vitamin_D\' AND n.id <> \'Vitamin_B_12\' AND n.id <> \'Zinc\' RETURN
n.id, r.quantidade ORDER BY n.id'
               resultNomeAlimento = graph_db.cypher.execute(str(query))
               for j in range(len(resultNomeAlimento)):
                 n = 0
                 while resultAlimentoIdeal[j][0] != nomesNutrientes[n]:
                   n += 1
                 somasNutrientes[n] += float(str(resultNomeAlimento[j][1]))
                 n += 1
               somaNutAtual = somasNutrientes[nut]
```