Projeto BD – Parte 3

Turno BDL03 – Grupo 32
Professor – Pedro Miguel Leão Veloso Dias

Realizado por:

André Santos – 99730
Diogo Miranda – 102536
David Pires – 103458

Percentagem de contribuição de cada aluno:

André Santos – 33.3%
Diogo Miranda – 33.3%
David Pires – 33.3%

Esforço Total:

12 horas (cada aluno)

1ª Parte: App

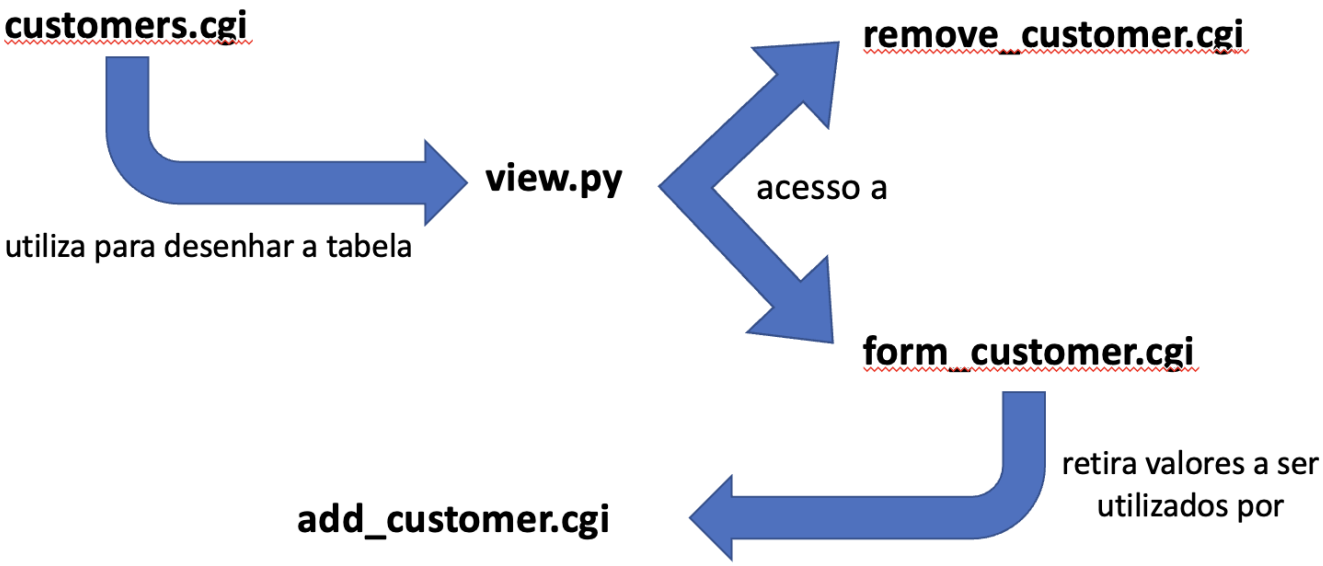
Link para a tabela customer: https://web2.tecnico.ulisboa.pt/ist1102536/customers.cgi

A aplicação web está dividida em 4 tabelas, nestas tabelas podemos ver os efeitos das operações pedidas no enunciado da entrega 3. Navegação entre tabelas pode ser feita no fundo da página ao clicar na tabela que quer aceder.

Os ficheiros com principais funções são os ficheiros add_*.cgi, remove_*.cgi, form_*.cgi, estes têm a lógica por detrás de cada ação, por exemplo add_customer.cgi contém a lógica por de trás de adicionar um cliente da tabela customer, os valores são retirados a partir de uma página á parte tratada pelo form_customer.cgi, remove_customer.cgi contém a lógica por de trás de remover um cliente da tabela customer.

Como todas as páginas com as tabelas têm a mesma estrutura, esta foi colocada num ficheiro em separado utilizado depois pelos .cgis das respetivas tabelas, o ficheiro view.py, que contém as funções view_table que dá print da tabela requisitada e os botões para as respetivas ações e a função handle_exception, que assegura que as páginas de erro são minimamente tratadas.

A ligação entre os seguintes ficheiros pode ser ilustrada através deste esquema como exemplo para a tabela customer.



2ª Parte: Índices

7.1.

SELECT order_no

FROM orders

JOIN contains USING (order_no)

JOIN product USING (SKU)

WHERE price > 50 AND

EXTRACT(YEAR FROM date) = 2023

- **Operação otimizada:**

"WHERE price > 50" - Filtragem de registos com base no valor da coluna "price".

- **Otimização:**

Criação de um índice na coluna "price" da tabela "product" que permitirá que o SGBD localize rapidamente os registos que atendem à condição "price > 50", de modo a melhorar o desempenho da filtragem.

- **Justificação:**

Criámos um index na tabela "product" sobre o atributo "price", pois temos uma procura que envolve operadores de comparação. Neste caso a procura em árvore é mais rápida, pois a coluna "price" tem valores numéricos que podem variar em magnitude, permitindo a rápida localização dos registos que atendem à condição "price > 50", utilizando o método BTREE que é capaz de organizar os valores em uma estrutura de árvore balanceada, reduzindo a quantidade de registos que precisam de ser percorridos. Assim, a fim de verificarmos isso, testámos num ficheiro sql de um populate com 9 milhões de linhas e reduziu o processo de 1962.899ms para 671.652ms, portanto reduziu o tempo, aproximadamente, em 65.82%.

- **Instrução de criação do índice SQL:**

CREATE INDEX product_price_idx ON product USING BTREE (price);

7.2.

SELECT order_no, SUM(qty*price)

FROM contains

JOIN product USING (SKU)

WHERE name LIKE ‘A%’

GROUP BY order_no;

- Operação otimizada:

"WHERE name LIKE 'A%'" - Filtragem de registos com base em um padrão de pesquisa na coluna "name".

- Otimização:

Criação de um índice na coluna "name" da tabela "product" que permitirá que o SGBD localize rapidamente os registos que atendem à condição "name LIKE 'A%'".

- Justificação:

Criámos um index na tabela "product" sobre o atributo "name", pois temos uma procura sequencial, o que em tabelas de dimensão elevada seria uma operação demorada. Neste caso a procura em árvore é mais rápida, pois o index BTREE permite que o banco de dados encontre rapidamente os registos cujos valores de "name" começam com 'A', reduzindo a quantidade de registos que precisam de ser percorridos.

- Instrução de criação do índice SQL:

CREATE INDEX product_name_idx ON product USING BTREE (name);