Relatório – Entrega 3

Contribuições

96925 | Gonçalo Silva | 20 Horas | 33.333%
99097 | José Cutileiro | 20 Horas | 33.333%
99113 | Miguel Vale | 20 Horas | 33.333%

Turno: L08
Professor: Miguel Silva
Número do Grupo: 062

0) Ficheiros

1. EsquemaBD.sql: Base de dados e RIs
2. populate.sql: Base de dados
3. ICs.sql: RIs
4. queries.sql: SQL
5. view.sql: Vistas
6. web/: Desenvolvimento da aplicação
7. analytics.sql: Consultas OLAP
8. relatorioV03: Índices

1) Arquivos (e inicializações)

1.1) Como correr a nossa Base de dados?

1. Esqueleto: \i EsquemaBD.sql
2. Restrições: \i ICs.sql
3. Vistas: \i view.sql
4. Dados: \i populate.sql

Notas: A primeira restrição de integridade, é mais sofisticada que o pedido no enunciado, evitando a circularidade entre categorias simples e super-categorias.

1.2) Consultas OLAP
As consultas OLAP estão no ficheiro analytics.sql

1.3) Consultas SQL
As consultas SQL estão no ficheiro queries.sql

2) Explicação da arquitetura da app

2.0 Aceder à nossa App?

Link: http://web.tecnico.ulisboa.pt/~ist196925/DB/app.cgi/
Link secundário: https://web2.ist.utl.pt/~ist199097/web/app.cgi/

2.1 Como usar a app?

Informações disponíveis:

1. Retalhistas presentes na nossa BD
2. Que categorias estão na nossa BD
3. Eventos de reposição na nossa BD
(agrupados por IVM e por categoria de produtos)

Ações disponíveis:

Eliminar Retalhista: Ao clicarmos na cruz que se encontra antes do nome do retalhista eliminamos o mesmo da nossa BD

Adicionar Retalhista: Ao clicarmos no botão "Adicionar Retalhista" irá aparecer um pop-up para escrevermos o nome do novo Retalhista, seguidamente esse mesmo nome irá aparecer na nossa BD

Visualizar Categorias: Ao clicarmos em cima de uma categoria conseguimos visualizar as suas categorias simples ou super categorias

Eliminar Categoria: Ao clicarmos na cruz que se encontra antes do nome da categoria eliminamos a mesma da nossa BD

Adicionar Categoria: Ao clicarmos no botão "Adicionar Categoria" irá aparecer um pop-up para escrevermos o nome da nova Categoria, seguidamente esse mesmo nome irá aparecer na nossa BD. Outro aspeto em ter em conta é quando adicionamos categorias dentro de uma categoria simples irá tornar a categoria simples numa super-categoria.

Selecionar IVM : Ao clicarmos no botão "Selecionar IVM" irá aparecer uma lista das IVM's disponiveis.

2.2. Desenvolvimento da app

2.2.1 Organização de ficheiros

A aplicação web recorre a uma arquitetura do tipo *Common Gateway Interface* (CGI), sendo constituída por uma aplicação Flask (app.cgi) e pela respetiva pasta de templates jinja. Dentro desta pasta encontra-se uma única template (index.html) responsável por qualquer interação entre o utilizador e o servidor (incluindo apresentação de erros que possam ocorrer).

2.2.2 Sintaxe URI

Todo o funcionamento da app baseia-se em parâmetros passados via URI juntamente com endpoints específicos a cada uma das ações:

Endpoint: (GET) /
Descrição: Página principal (retalhistas, categorias, eventos)
Parâmetros disponíveis:

1. cat: Localização da categoria atual na "árvore de categorias" (i.e. /Bebidas/Aguas)
2. ivm: Identificação da ivm onde se quer visualizar eventos. Usa-se o formato num_serie-responsavel (i.e. 5-Maria Domingas)
3. errors (apenas uso interno): Permite que uma ação consiga comunicar erros à interface de utilizador

Exemplo: app.cgi/?cat=/Bebidas/Aguas&ivm=5-Maria%20Domingas

Endpoint: (POST) /add_cat
Descrição: Adiciona uma sub-categoria à categoria fornecida (cat)
Parâmetros disponíveis:

1. cat: Localização da categoria na "árvore de categorias" (i.e. /Bebidas/Aguas)
2. *(application/x-www-form-urlencoded)* categoria: Nome da nova categoria

Exemplo: app.cgi/add_cat?cat=/Bebidas/Aguas/

Endpoint: (GET) /delete_cat
Descrição: Remove a categoria fornecida (cat) bem como as relações que esta tem com outras categorias
Parâmetros disponíveis:

1. cat: Localização da categoria na "árvore de categorias" (i.e. /Bebidas/Aguas)

Exemplo: app.cgi/delete_cat?cat=/Bebidas/Aguas/Vitalis

Endpoint: (POST) /add_ret
Descrição: Adiciona um novo retalhista
Parâmetros disponíveis:

1. *(application/x-www-form-urlencoded)* retalhista: Nome do retalhista a adicionar

Exemplo: app.cgi/add_ret

Endpoint: (GET) /delete_ret
Descrição: Remove o retalhista fornecido (ret) bem como as relações que este tem com os produtos e categorias
Parâmetros disponíveis:

1. ret: Nome do retalhista a remover (i.e. Samsang)

Exemplo: app.cgi/delete_ret?ret=Samsang

2.2.3 DBClient

Esta classe criada tem como principal objetivo simplificar a execução de queries seguras na base de dados bem como evitar a repetição de código.

- **DBClient.queryWrapper(query, values)**
  Permite efetuar uma query com ou sem valores a serem concatenados (values=None)
  Executa cursor.fetchall no caso de se tratar de uma query com SELECTs
  NÃO termina a ligação à base de dados, sendo por isso necessário terminar manualmente com DBClient.close()

  Exemplo:
  db_client.queryWrapper("INSERT INTO categoria VALUES (%s)", ("sobremesas",))

- **DBClient.renderPageForQueries(page, queries)**
  Efetua o render da template fornecida (page) juntamente com um dicionário (ver em baixo exemplo de estrutura) que define todos os contextos a que a template tem acesso.
  Termina a ligação à base de dados após gerar a página.

```
queries = {
        "query_simples": "SELECT ... FROM ...",
        " query_com_valores": {
                "query": "SELECT ... FROM ... WHERE … = %s AND … = %s",
                "values": ("value1", "value2", )
        },
        "valor_a_ser_exposto": {
                "query": "CONTEXT_NOT_QUERY",
                "values": "value1"
        }
}
```

- **DBClient.commit()**
  Executa commit das operações

- **DBClient.close()**
  Termina a ligação com a base de dados

3) Índices

3.1 Objetivo:

Saber o nome de quais os retalhistas são responsáveis por uma dada categoria (i.e 'Frutos')

Query:

```
SELECT DISTINCT R.nome
FROM retalhista, resposavel_por P
WHERE R.tin = P.tin and P.nome_cat = 'Frutos'
```

Que índice usar:

1. O índice a utilizar deve ser não ordenado
dado nenhuma das colunas das nossas tabelas tem uma
ordem especificada.

2. O índice a utilizar deve ser denso dado que
os esparsos só podem ser utilizados caso as colunas estejam
ordenadas de alguma forma.

3. Os índices de dispersão são os melhores para selecionar
entradas por igualdade. Neste exemplo estamos a selecionar
os nomes dos retalhistas que são responsáveis pelas categorias
IGUAIS a 'Frutos'. Dada a igualdade o índice escolhido deverá ser
um índice de dispersão. Deverá também ser dinâmica dado que o nº de
contentores pode variar ao longo do ciclo de vida do índice.

Conclusão:

O índice deve ser:
- Não ordenado
- Denso
- Dispersão dinâmica

Criar índice:

```
CREATE INDEX t_idx on responsavel_por USING HASH(nome_cat);
```

3.2 Objetivo:

Contar quantos produtos estão associados a cada categoria e tem uma descrição num dado formato (i.e. Começar pela letra A)

Query:

```
SELECT T.nome, COUNT(T.ean)
FROM produto P, tem_categoria T
WHERE p.cat = T.nome AND P.desc LIKE 'A%'
GROUP BY Y.nome
```

Que índice usar:

1. O índice a utilizar deve ser não ordenado dado nenhuma das colunas das nossas tabelas tem uma ordem especificada.

2. O índice a utilizar deve ser denso dado que os esparsos só podem ser utilizados caso as colunas estejam ordenadas de alguma forma.

3. A chave de procura deverá ser composta dado que as interrogações requerem várias comparações (Nome da categoria e Descrição associada ao produto)

4. Deverá ser utilizado um Índice do tipo B+, mesmo que tenhamos um número muito grande de entradas selecionadas, é possível chegar rapidamente aos objetivos e com um custo de memória adicional bastante diminuto. Se houver algum UPDATE na tabela, basta fazermos pequenas alterações locais no índice para que este funcione novamente.

Conclusão:

O índice deve ser:
- Não ordenado
- Denso
- B+

Criar índice:

```
CREATE INDEX f_idx on produto(cat,descr);
```

4) Triggers – Notas adicionais

Nota inicial: Os triggers estão no ficheiro ICs.sql. Estão implementados com um sistema de mensagem facilmente identificável caso o utilizador tente ultrapassar as barreiras do domínio.

Nota RI-1: Segundo o domínio esta restrição devia evitar circularidade nas relações entre categorias. Contudo já no final do projeto um dos professores disse que não era preciso este nível de preocupação e que apenas precisamos de evitar as relações diretas. Apesar disso como o grupo já tinha dedicado tempos significativos para fazer a concretização original como era pedido inicialmente, decidimos manter assim.