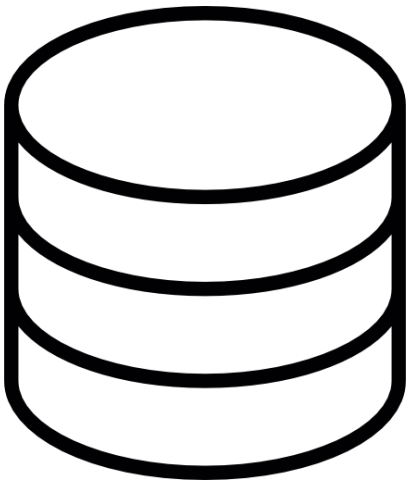
Projeto BD - Parte 3

Francisco Gouveia (102571), Rui Amaral (103155) e Pedro Freitas (103168)



Turno Laboratorial: L03

Professor Responsável pelo Laboratório: Pedro Leão Dias

Site: https://web2.tecnico.ulisboa.pt/ist1102571/main.cgi

Número do Grupo: 09

Percentagem Relativa de Contribuição de cada aluno:
-> Francisco Gouveia = 33%
-> Rui Amaral = 33%
-> Pedro Freitas = 34%

Esforço Total (em horas) de cada aluno:
-> Francisco Gouveia = 25h
-> Rui Amaral = 25h

-> Pedro Freitas = 25h

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
```

```
CREATE INDEX order_date ON orders USING BTREE(EXTRACT(YEAR
FROM date), order_no);
```

Este índice permitirá a filtragem eficiente dos registros da tabela `orders` pela condição `EXTRACT(YEAR FROM date) = 2023`. Decidimos criar um índice composto com o order_no para permitir um acesso index-only. Deste modo, o sistema consegue encontrar os `order_no` das `orders` com data em 2023 rapidamente e sem precisar aceder à própria tabela orders. Foi por esta razão que optamos por um índice BTREE, visto que com um índice HASH, apesar da procura pelos registos que respeitam a condição `EXTRACT(YEAR FROM date) = 2023` ser mais rápida, não podíamos ter um index-only scan pois seria necessário aceder à tabela orders para obter o `order_no` de cada registo.

```
CREATE INDEX prod_price ON product USING BTREE(price, SKU);
```

Este índice facilita a procura pelos produtos com `price > 50`. De forma análoga à explicação anterior, incluímos também o SKU no índice do modo tornar possível um index-only scan. Assim, o sistema é capaz de encontrar os SKUs de todos os produtos com preço maior que 50 de forma eficiente e sem fazer algum acesso à tabela orders.

Não vimos necessidade de criar novos índices para facilitar as operações de `JOIN` devido à existência de índices de primary keys das tabelas envolvidas que podem ser usados para esse propósito (`contains(order_no, SKU)`, `product(SKU)` e `orders(order_no)`). Além disso, a criação de índices implica uma perda de performance significativa para operações básicas da BD, por isso, tentámos limitar ao máximo o número de índices redundantes na base de dados.

7.2.

```
SELECT order_no, SUM(qty*price)
FROM contains
JOIN product USING (SKU)
WHERE name LIKE 'A%'
GROUP BY order_no;
```

```
CREATE INDEX prod_name ON product USING BTREE(name
text_pattern_ops, price, SKU);
```

Para esta query, implementamos um índice BTREE no `name` da tabela produto. Como o filtro no nome é uma procura por um prefixo (nomes a começar por "A"), podemos facilitar essa procura com um índice BTREE. No índice, incluímos também o `price` e o SKU do produto de forma a permitir um index-only scan que irá obter toda a informação relevante dos produtos com nome a começar por "A" de forma eficiente e sem acessos à tabela `product`.
O operador `text_pattern_ops` foi incluído de modo a tornar este índice útil a filtros do tipo `LIKE` ou `POSIX` no nome dos produtos.

Não achámos necessária a criação de mais índices, pois, para as outras operações (agregação e `JOIN`), o sistema tem à sua disposição os índices das primary keys das tabelas `contains` (que prioritiza o `order_no` e portanto pode ser utilizado para uma agregação eficiente) e `product`. É também importante relembrar de que há um preço (a nível de performance) a pagar por se adicionarem novos índices à base de dados como já foi aqui referido.

Durante os nossos testes, reparámos que o sistema muitas vezes não reagia como havíamos previsto segundo a teoria ensinada em aula. Muitas vezes, os índices criados nem chegavam a ser utilizados, e, quando eram, acabavam por ter pouco impacto na performance das queries. Supomos que isto seja devido ao facto de os nossos dados serem sintéticos, e apresentarem algum "bias" que acaba por implicar um comportamento atípico por parte do planeador do Postgres. Ainda assim, decidimos seguir as regras teóricas para a implementação de índices.

Aplicação Web

Dentro da pasta web temos vários tipos de ficheiros, sendo do tipo “cgi”, “py” ou “css”. Temos no entanto apenas um ficheiro de styling, “styles.css”, este ficheiro trata do design de toda a aplicação web e é importado em cada página web apresentada ao utilizador. Depois temos 3 ficheiros python, um “secret.py” que contém a palavra-passe para a base de dados, um “login.py” que tem as credenciais para acesso à base de dados, e por fim um ficheiro “aux.py”. Este último ficheiro apenas tem duas funções cujo objetivo é evitar a repetição de código em todos os ficheiros “.cgi”, executando apenas “prints” de snippets de HTML.

Olhando agora para a arquitetura da aplicação, temos um ficheiro inicial, que representa a “home page” da nossa app. Esse ficheiro é o “main.cgi” que contém 6 botões de modo a permitir ao utilizador escolher que ação pretende fazer. As ações possíveis, representadas pelos botões, são as seguintes: “Pay an Order”, “Execute an Order”, “Client Management”, Update a Product”, “Product Management”, “Supplier Management”. Todas as ações exceto as que terminam em “Management” vão dar a uma outra página, respetivamente gerada pelo seu ficheiro “.cgi” em que apresentam um “form” ao utilizador de forma a poder receber input e depois executar uma ação com ele. De notar, que todas as ações têm sempre a opção para voltar ao menu inicial, de modo a que o utilizador tenha sempre uma forma de voltar ao ponto de partida.
Agora, em relação às ações que terminam em “Management”, estas são simplesmente um outro sub-menu que dependendo das ações das quais foram chamadas, apresentam um menu de dois botões, um de “Create” e outro de “Delete”, seja para “Supplier”, “Product” ou “Customer”. Isto foi feito com o intuito de diminuir a complexidade e simplificar o primeiro menu que era apresentado ao utilizador.

Depois, mais à frente, cada página acaba por ter um “form” em que é possível passar informação para o back-end, onde depois, a utilizamos para executar, “queries”, “deletes”, “updates” ou “creates”. A solução da aplicação web apresentada pelo grupo teve em consideração o ataque “SQL-Injection” tendo-se protegido deste.