Projeto BD – Parte 3

2º Semestre – 2022/2023

Grupo 27

Nº aluno	Nome	Percentagem de contribuição	Esforço	Turno	Professor
102879	António Silva	33,33%	11:00	BD2L06	Tomás Caldeira Pedro Sousa
102760	Diogo Marques	33,33%	11:00		
103215	Leonardo Pinheiro	33,33%	11:00		

1. Carregamento da Base de Dados

O populate.sql permite o preenchimento da base de dados de forma consistente, recorrendo a funções que preenchem de forma aleatória as tabelas, bem como a inserção multi-valor.

2. Restrições de Integridade

Foram implementadas as Restrições de Integridade descritas no enunciado sob a forma de *Stored Procedures* e *Triggers* no ficheiro **ICs.sql**. As RI-1, RI-2 e RI-3 são concretizadas respetivamente pelos *before triggers* **trigger_age_of_an_employee** e **trigger_type_of_workplace** e pelo *after trigger* **trigger_order_in_contains**, que, respetivamente, invocam os *stored procedures* **chk_age_of_an_employee**, **chk_type_of_workplace** e **chk_order_in_contains**.

3. SQL

De modo a realizar as *queries* pedidas, aplicámos a matéria lecionada nas aulas. A primeira *querie*, utiliza uma *nested querie* de forma a obter o valor máximo entre todas as encomendas pagas. Já na segunda, foi utilizada a abordagem da divisão de forma a obter todas as encomendas realizadas no ano de 2022 e depois filtrar dentro dessas, as datas das encomendas que foram realizadas no ano de 2022 e não foram processadas pelo funcionário que está a ser verificado, se não for retornado nenhum resultado, isso significa que o funcionário processou encomendas em todos os dias de 2022 em que houve encomendas.

4. Vistas

Para a *view* foi feito o *join* das tabelas com as informações necessárias/pedidas, bem como a renomeação de algumas colunas.

5. Desenvolvimento/Arquitetura da aplicação Web

É possível encontrar uma versão do trabalho, através do link abaixo:
http://web2.tecnico.ulisboa.pt/ist1102760/home_page.cgi

Na página inicial encontra-se um menu para consultar as diferentes opções pedidas. Através da aba "CUSTOMER MANAGER" é possível **registar novos clientes, assim como removê-los**. O registo de novos clientes requer o preenchimento de cinco formulários referentes ao cust_no, *name*, *email*, *phone* e *address*. Enquanto que para remover o cliente basta preencher o formulário com o cust_no, uma vez que este é a *primary key* da tabela *customer*. A remoção de um cliente implica automaticamente a remoção das suas encomendas, bem como das suas vendas, logo são apagadas todas as encomendas da tabela *order* e *pay* com o cust_no do cliente removido.

Através da aba "SUPPLIER MANAGER" é possível **registar novos fornecedores e produtos, assim como removê-los**. O registo de novos fornecedores permite ainda a criação de novos produtos ou a associação de produtos existentes a novos fornecedores. Qualquer uma destas opções requer o preenchimento de nove formulários referentes ao TIN, *name*, *address*, SKU do produto, *date*, nome do produto, descrição do produto, preço do produto e o código EAN. Para remover o fornecedor, basta preencher o formulário com o TIN, uma vez que este é a *primary key* da tabela *supplier*. Ao remover-se um fornecedor, caso esse fornecedor seja o único que fornece um produto x, esse produto é igualmente apagado, caso contrário o fornecedor é simplesmente apagado.
Através da aba "PRODUCT MANAGER" é possível **alterar o preço e descrição e ainda remover produtos**. Qualquer alteração na tabela *product*, assim como a remoção requer o preenchimento de um formulário com o SKU, uma vez que este é a *primary key*. As alterações requerem ainda a nova alteração a fazer. Quando à remoção de um produto implica também a remoção de todas as encomendas que contêm esse produto, assim como a remoção de todos os fornecedores desse produto.
Através da aba "ORDER MANAGER" é possível **realizar uma encomenda, assim como pagar a mesma**. De forma, a realizar uma encomenda é necessário preencher cinco formulários referentes à order_no, ao cust_no do cliente que realiza a mesma, o SKU do produto que é encomendado, quantidade e a data da encomenda. Para pagar é necessário apenas o order_no e cust_no. Ao realizar-se um pagamento de uma encomenda, esta é adicionada à tabela *pay*, no entanto mantém-se na tabela *order*.
A aplicação é gerida através do **home_page.cgi**, uma vez que é a partir deste ficheiro que se vai obter a redirecção para os restantes ficheiros e consequentes alterações à base de dados. Após qualquer operação na base de dados é apresentada uma mensagem de sucesso, caso seja realizada a operação e volta-se ao ponto inicial: **home_page.cgi**.
De forma a prevenir a aplicação de qualquer tipo de SQL INJECTION todos os parâmetros das queries têm um nome.

6. Consultas OLAP

Para a análise do valor total das vendas foi utilizado o mecanismo ROLLUP para obter os resultados especificados no enunciado. Para a primeira *query*, foi utilizada, a função SUM e o ano de 2022. Do mesmo modo, para a segunda *query*, foi utilizada a função AVG.

7. Indíces

7.1.

Na operação JOIN contains USING (order_no), o order_no já é uma primary key e uma foreign key da tabela contains (order_no, SKU), logo não justifica criar outro indíce. O mesmo acontece na linha seguinte - JOIN product USING (SKU) - onde SKU é uma primary key e não é necessário por isso criar um índice.
Na parte de filtragem da *query* - WHERE price > 50 AND EXTRACT (YEAR FROM date) = 2023 - podemos criar dois índices de forma a agilizar a execução da consulta. No caso de *price* seria um índice do tipo **B-TREE**, pois este índice facilita na procura de

mais que um item, uma vez que se encontram todos à mesma distância da raíz. Já para *date* seria um índice do tipo **HASH**, dado que este é o tipo ideal quando temos igualdades, como neste caso.

```
CREATE INDEX idx_product_price ON product USING BTREE (price);
CREATE INDEX idx_orders_date ON orders USING HASH (date);
```

7.2.

Na operação JOIN product USING (SKU), como SKU já é uma primary key da tabela product não será necessário criar índice, pois ele é criado implicitamente quando declaramos a primary key.
Na linha WHERE name LIKE 'A%' já podemos criar um índice de forma a agilizar a consulta. Sendo esse índice do tipo B-TREE para *name* na tabela *product*, e assim poderíamos ter uma otimização na altura de filtrar o nome dos produtos.
No que diz respeito à linha GROUP BY order_no, não precisamos de criar um índice pois como order_no é uma primary key da tabela orders já tem um índice.

```
CREATE INDEX idx_product_name ON product USING BTREE(name);
```