Universidade de Trás-os-Montes e Alto Douro

Fase 3 – Protocolo E

Licenciatura em Engenharia Informática

Base de Dados

Paulo Martins – pmartins@utad.pt
Daniel Alexandre – daniel@utad.pt

Autores
Eduardo Manuel Afonso Chaves - 70611
João Henrique Constâncio Rodrigues - 70579
Luís André de Marques Pimenta - 70827

Vila Real, junho de 2021

Índice

1. Introdução

Neste relatório, iremos introduzir certos conceitos de modo a apresentar os assuntos retratados na unidade curricular de Base de Dados, com um objetivo inicial de apresentar todo o conhecimento adquirido em Base de Dados. Conceitos esses relacionados com a Normalização, Arquitetura de base de dados e linguagem SQL.

Com esta 3º fase do trabalho tivemos em conta mais alguns comandos da linguagem SQL. Com o código com população e ainda com algumas funções já implementadas tivemos que ir mais a fundo e introduzir mais dois conceitos de SQL para poder resolver os exercícios propostos. Conceitos esse como stored *procedure* e *trigger*.

2. Enquadramento teórico

2.1. Linguagem SQL

Para entendermos um pouco do que se trata a linguagem SQL vamos partir dos comandos mais básicos que foram usados na primeira e segunda parte do projeto. Em primeiro são criadas a base de dados e as suas tabelas de forma a estruturar o esqueleto do nosso projeto. É dado tipos aos dados que serão colocados nas tabelas (ex: **INTEGER**, **CHAR**, **DATE** e **TIME**), além disso temos que especificar se as colunas obtêm ou não valores nulos. Comandos necessários:

- ➔ **CREATE** - Comando necessário para a construção de uma base de dados (**CREATE DATABASE**) ou criar uma tabela (**CREATE TABLE**);
- ➔ **DROP** - Permite remover uma base de dados (**DROP DATABASE**) ou remover uma tabela (**DROP TABLE**).

De seguida depois de tudo criado temos que inserir valores, neste caso denominados de registos, para podermos testar a base de dados. Isso só será possível a partir dos comandos **INSERT INTO** para especificar a tabela e **VALUES** para especificar os resultados. Com as tabelas criadas podemos a qualquer momento aceder aos dados das tabelas, criando assim tabelas novas a partir de outras já existentes, recorrendo ao **SELECT**. Assim de formamos uma base de dados no contexto do trabalho atribuído e de forma geral esses são os comandos mais utilizados nestas duas fases do trabalho

Existe também outros comandos que temos de ter em consideração como o **UPDATE** que permite alterar os valores já existentes nos campos de uma única tabela; o **WHERE** para selecionar um conjunto de linhas; o **AS** serve para renomear uma coluna ou uma tabela ; **MAX/MIN/TOP** para devolver o maior ou menor valor ou o valor do topo da coluna; **COUNT** serve para devolver o números de linhas na qual foi especificado; **GROUP BY** serve para agregar funções; **ORDER BY** para ordenar de forma ascendente (**ASC**) ou descendente (**DESC**); o **CAST** é uma função que converte um valor de qualquer tipo em um tipo especifico; o **GETDATE** para recebermos a data e hora no exato momento em que corremos a função em formato 'YYYY-MM-DD hh:mm:ss.mmm'; o **DATEDIFF** retorna a diferença entre duas datas.

Por último, iremos falar da SubQuerys que no fundo é uma query dentro de uma query maior, ou de certa forma um **SELECT** dentro de outro. As SubQuerys intervêm de modo a tornar certas consultas mais simples que de outra forma seriam completamente complicadas ou impossíveis.

Nesta 3º fase do trabalho introduzimos mais dois comandos na implementação dos exercícios. Começamos por introduzir as ***Stored Procedures***, que é um conjunto de comandos que podem ser executados como uma função, ou seja, ele armazena um conjunto de regras que podem ser usados de acordo com a necessidade individual. Eles surgem com o objetivo de melhorar a performance de um banco de dados tornando tudo mais simples e rápido.

De seguida temos os ***Triggers***, é uma estrutura definida numa base de dados que funciona como se fosse uma função que efetua uma ação quando há alguma operação. Este comando está diretamente relacionado a uma tabela, então sempre que é efetuado uma operação nessa tabela é disparado o "*trigger*" para executar alguma tarefa em específico.

3. Desenvolvimento do trabalho

3.1. Criação dos procedimentos e do trigger

```
use master;
USE Presidenciais;

----------------------ALL SELECT-----------------------
SELECT * FROM Pessoas;
SELECT * FROM Candidatos;
SELECT * FROM Escritorios;
SELECT * FROM Candidatos_Escritorios;
SELECT * FROM Presidentes;
SELECT * FROM Vogais;
SELECT * FROM Descricao;
SELECT * FROM Cargos;
SELECT * FROM Mesa_eleitoral;
SELECT * FROM Orcamento;
SELECT * FROM Candidatura;
SELECT * FROM Mandatario;
SELECT * FROM Local_votar;
SELECT * FROM Votar;
SELECT * FROM Numero_votos;
SELECT * FROM Assumir;
SELECT * FROM Presidir;
SELECT * FROM Hora_participar;
SELECT * FROM Participar;
-------------------------------------------------------

--1.Crie um procedimento que, dados o ID de uma Mesa Eleitoral e o
Mês, apresente uma tabela com os pares de vogais que participaram
--em cada Mesa Eleitoral. O procedimento deve devolver o número
--total de pares distintos que participaram nas mesas eleitorais.

CREATE PROCEDURE Presencas_Vogais(@Mesa_ID INT, @Mes INT)
AS
BEGIN
SELECT titulo AS Mesa, Pessoas.nome, SQ1.nome
FROM Participar, Vogais, Pessoas, Mesa_eleitoral, (
       SELECT MIN(data_participar) AS min_data_participar, nome, id_mesa_eleitoral
       FROM Participar, Vogais, Pessoas
       WHERE Pessoas.numero_eleitor = Vogais.numero_vogal
       AND Vogais.numero_vogal = Participar.numero_vogal_b
       GROUP BY nome, id_mesa_eleitoral)SQ1
WHERE Pessoas.numero_eleitor = Vogais.numero_vogal
AND Vogais.numero_vogal = Participar.numero_vogal_a
AND Participar.id_mesa_eleitoral = Mesa_eleitoral.id_mesa_eleitoral
AND Participar.data_participar = SQ1.min_data_participar
GROUP BY titulo, Pessoas.nome, SQ1.nome
ORDER BY titulo

SELECT COUNT(DISTINCT(numero_vogal_a + numero_vogal_b)) AS N_Total_Pares
FROM Participar
--WHERE id_mesa_eleitoral = @Mesa_ID
--AND DATEPART(MONTH, data_participar) = @Mes
```

```
END
--DROP PROCEDURE Presencas_Vogais

--Exemplo:
EXECUTE Presencas_Vogais 3, 5

-----------------------------------------------------

--2.Assumindo que todas as pessoas com mais de 18 anos podem votar,
crie um procedimento que dado o número de eleitor do candidato, o
--cargo e a data da candidatura, para uma votação em curso, verifique
-- a percentagem de pessoas que já participaram na votação.

CREATE PROCEDURE Percentagem_Votos(@NCandidato INT, @Cargo
VARCHAR(10), @DataCandidatura DATE)
AS
BEGIN
SELECT ((COUNT(Votar.numero_eleitor)*100)/(SELECT COUNT(*) FROM
Pessoas)) AS Percentagem
FROM Pessoas, Votar, Candidatura, Cargos, Candidatos
WHERE Pessoas.numero_eleitor = Votar.numero_eleitor
AND Candidatos.numero_candidato = @NCandidato
AND Cargos.titulo LIKE @Cargo
AND Candidatura.data_candidatura = @DataCandidatura
AND DATEDIFF(YY, Pessoas.data_nascimento, GETDATE()) >= 18
AND data_votar <= GETDATE()
END
--DROP PROCEDURE Percentagem_Votos

--Exemplo:
EXECUTE Percentagem_Votos 2, 'Tesoureiro', '2021-01-24'

-----------------------------------------------------

--3.Assumindo que uma Pessoa não pode assumir dois cargos em simultâneo,
crie um trigger que ao inserir um registo de um cargo para um candidato
--na tabela assumir automaticamente insira a mesma data como data de
-- fim no cargo que o candidato anteriormente ocupava.

CREATE TRIGGER NewTrigger
ON Assumir
AFTER INSERT
AS
BEGIN
      DECLARE @num_Candidato  INTEGER,
              @data_Inicio    DATE

      SELECT @num_Candidato = numero_candidato, @data_Inicio = data_inicio
      FROM inserted
      IF((SELECT COUNT(@num_Candidato) FROM Assumir) > 1)
      BEGIN
            UPDATE Assumir
            SET data_fim = @data_Inicio
            WHERE numero_candidato = @num_Candidato
```

```
106                 AND data_fim IS NULL
107                 AND data_Inicio < @data_Inicio
108           END
109     END
110     --DROP TRIGGER NewTrigger
111
112     --Exemplo:
113     SELECT * FROM Assumir;
114     INSERT INTO Assumir(id_cargos, numero_candidato, data_inicio)
115     VALUES (2, 6,'2021-06-10');
116     SELECT * FROM Assumir;
```

4. Conclusão

Com todos os conceitos trabalhados e tratados ao longo do relatório podemos então chegar à conclusão desta segunda etapa do trabalho.

Com a terceira fase aprendemos que é possível aumentar a funcionalidade de uma base de dados acedendo a funções como Stored Procedures e Triggers, já que estes comandos podem aumentar a performance de uma base de dados em termos gerais. Isto é extremamente vantajoso pois proporciona uma base de dados mais rápida e coesa.

5. Bibliografia

- Paulo Martins, (2021). Conceção e Desenvolvimento de Bases de Dados v2
- Paulo Martins, (2021). Linguagem SQL v2