

Base de Dados 2 (BD2)

Trabalho de Avaliação

Elaborado por:

Fábio Gonçalves nº 17646

João Portelinha nº 20481

Docentes:

Isabel Brito

Gonçalo Fontes

27/01/2021

Índice

Índice de Figuras	2
Introdução.....	4
Modelo Físico	5
Criação da Base de Dados e Gestão dos seus Elementos	6
Criação da Base de Dados e Gestão dos seus Elementos (Continuação)	7
Criação da Base de Dados e Gestão dos seus Elementos (Continuação 2)	8
Criação das Tabelas	9
Preenchimento da Base de Dados	10
Preenchimentos da Base de Dados (Continuação).....	11
Preenchimento da Base de Dados (Continuação 2)	12
Stored Procedures.....	13
Stored Procedures (2)	14
Stored Procedures (3)	15
Stored Procedures (4)	16
Stored Procedures (5)	17
Stored Procedures (6)	18
Stored Procedures (7)	19
Stored Procedures (8)	20
Triggers.....	21
Triggers (2)	22
Triggers (3)	23
Medidas de Segurança	24
Cópias de Segurança	25
Cópias de Segurança (2)	26
Cópias de Segurança (3)	27
Cópias de Segurança (4)	28
Cópias de Segurança (5)	29
Cópias de Segurança (6)	30
Cópias de Segurança (7)	31
Cópias de Segurança (8)	32
Cópias de Segurança (9)	33
Conclusão	34
Bibliografia	35

Índice de Figuras

Figura 1 - Ficheiro onde se encontra a criação da Base de Dados e da criação dos respetivos Discos e FileGroups	4
Figura 2 - Ficheiro onde se encontra a criação das Tabelas da Base de Dados	4
Figura 3 - Ficheiro onde se encontram os Dados inseridos nas Tabelas.....	4
Figura 4 - Ficheiro onde se encontram as Procedures.....	4
Figura 5 - Ficheiro onde se encontram os Triggers	4
Figura 6 - 3 Discos	4
Figura 7 - Diagrama Modelo Relacional (SQL)	5
Figura 8 - Primary Filegroup.....	6
Figura 9 - Covid_Filegroup_Tables.....	6
Figura 10 – Covid19_Filegroup_Tables_2	7
Figura 11 - Covid19_Filegroup_Tables_3.....	7
Figura 12 - Covid19_Filegroup_Index	8
Figura 13 - Organização dos Logs.....	8
Figura 14 - Criação das Tabelas.....	9
Figura 15 - Preenchimento de Tabelas simples	10
Figura 16 - Exemplo de Preenchimento de Tabelas Concelhos e Cidades.....	10
Figura 17 - Ferramenta utilizada para gerar Dados para a tabela Pessoas.....	11
Figura 18 - Inserção de dados na tabela Países	12
Figura 19 - Inserção de dados na tabela Regioes.....	12
Figura 20 - sp_DadosPortugal	13
Figura 21 - Resultados sp_DadosPortugal	13
Figura 22 - sp_DadosGE	14
Figura 23 - sp_DadosConcelhos	14
Figura 24 - Resultado sp_DadosGE Figura 25 - Resultado sp_DadosConcelhos	14
Figura 26 - sp_Concelho.....	15
Figura 27 - Resultado sp_Concelho.....	15
Figura 28 - sp_ConcelhoCidades	16
Figura 29 - Resultados sp_ConcelhoCidades	16
Figura 30 - sp_TotalGenero.....	17
Figura 31 - sp_AtivosRegiao	17
Figura 32 - Resultados sp_TotalGenero Figura 33 - Resultados sp_AtivosRegiao ...	17
Figura 34 - sp_RecuperadosRegiao.....	18
Figura 35 - sp_ObitosRegiao	18
Figura 36 - sp_ConfirmadosRegiao	18
Figura 37 - sp_MortesPorIdade	19
Figura 38 – sp_AtivosPorPais	19
Figura 39 - Resultados "sp_MortesPorIdade" Figura 40 - Resultados "sp_AtivosPorPais"	19
Figura 41 - sp_RecuperadosPorPais.....	20
Figura 42 - sp_ObitosPorPais	20
Figura 43 - sp_ConfirmadosPorPais	20

Figura 44 - tr_Dados24h	21
Figura 45 - Tabela "Dados24h"	21
Figura 46 - tr_DadosTotal	22
Figura 47 - Tabela "DadosTotal"	22
Figura 48 - tr_Seguranca	22
Figura 49 - tr_AlertaPopulacao	23
Figura 50 - Resultado "tr_AlertaPopulacao"	23
Figura 51 – Roles	24
Figura 52 - Users	24
Figura 53 - Permissões	24
Figura 54 - Passo 1	25
Figura 55 - Passo 2	25
Figura 56 - Passo 3	26
Figura 57 - Passo 4	26
Figura 58 - Passo 5	27
Figura 59 - Passo 6	27
Figura 60 - Passo 7	28
Figura 61 - Passo 8	28
Figura 62 - Passo 9	29
Figura 63 - Passo 10	29
Figura 64 - Passo 11	30
Figura 65 - Passo 12	30
Figura 66 - Passo 13	31
Figura 67 - Passo 14	31
Figura 68 - Passo 15	32
Figura 69 - Passo 16	32
Figura 70 - Passo 17	33

Introdução

O presente relatório tem origem da realização do projeto de avaliação da cadeira de Base de Dados 2, em que temos como objetivo fazer uma simulação o mais realista possível de como é toda a estrutura das estatísticas mostradas pela DGS em que mostra o estado “atual” (ou o mais perto disso) sobre a pandemia.

Com este projeto então pretendemos demonstrar isso fazendo a sua criação de base de dados e dos seus respetivos conteúdos, incluindo o seu **método de preenchimento**, assim como, **stored procedures** e **triggers** que serão utilizados na mesma, ao fazendo também a sua respetiva **segurança** e **cópias de segurança** como **backups** da base de dados e por fim nos últimos passos do relatório iremos também mostrar aspetos do seu **desempenho** e a sua **manutenção** e **automatização do servidor**.

Em anexo a este relatório encontram-se os seguintes ficheiros que se encontram nas figuras abaixo:

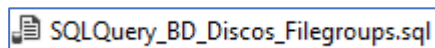


Figura 1 - Ficheiro onde se encontra a criação da Base de Dados e da criação dos respetivos Discos e FileGroups

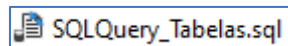


Figura 2 - Ficheiro onde se encontra a criação das Tabelas da Base de Dados

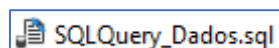


Figura 3 - Ficheiro onde se encontram os Dados inseridos nas Tabelas

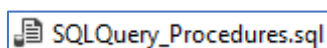


Figura 4 - Ficheiro onde se encontram as Procedures

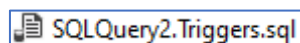


Figura 5 - Ficheiro onde se encontram os Triggers

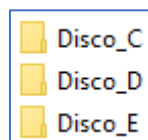


Figura 6 - 3 Discos

Modelo Físico

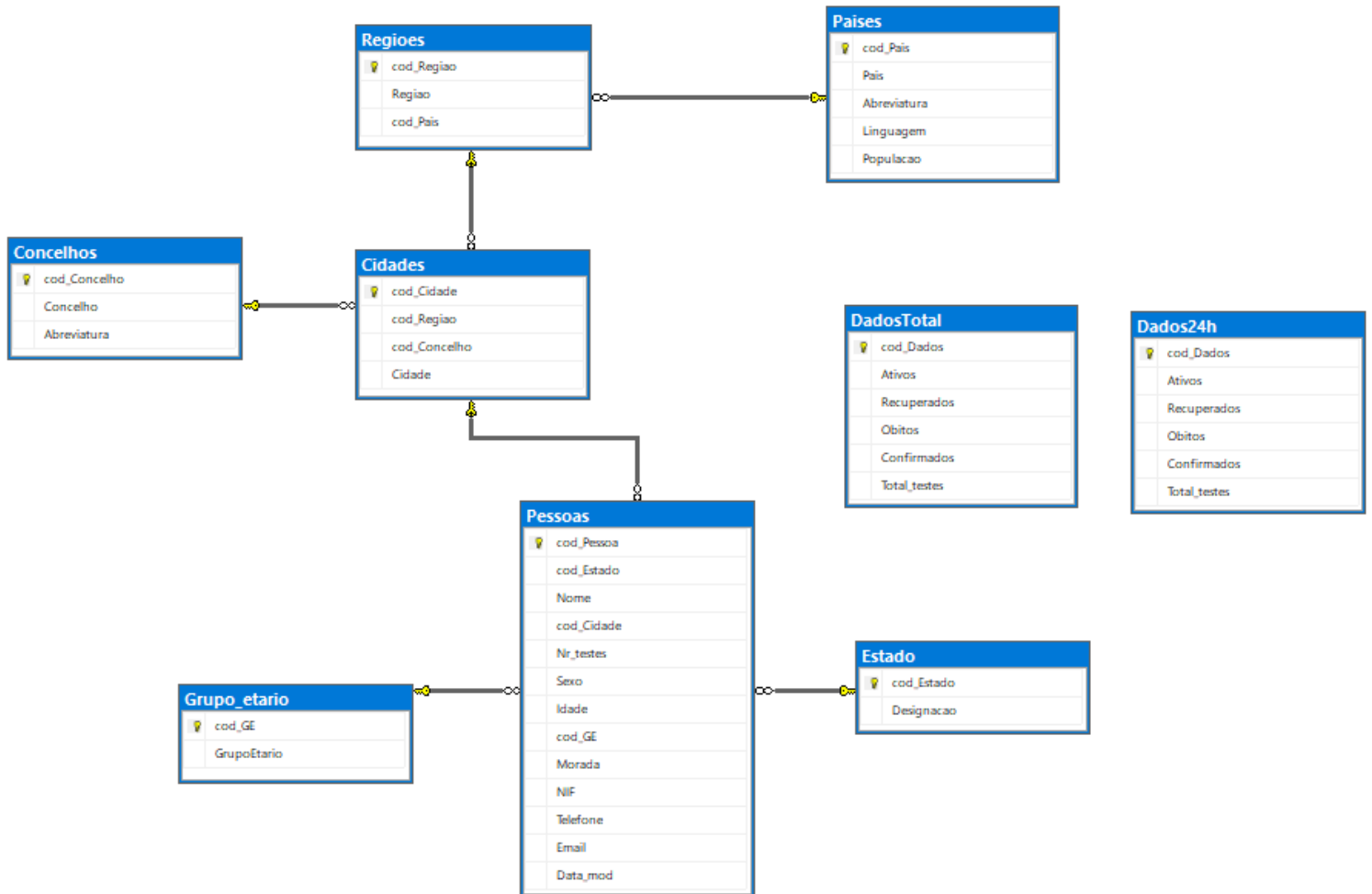


Figura 7 - Diagrama Modelo Relacional (SQL)

Criação da Base de Dados e Gestão dos seus Elementos

Neste tópico serão descritos todos os procedimentos que foram realizados em ordem para a criação e gestão da base de dados e de todos os seus elementos.

Primeiramente decidimos criar a base de dados com o nome “Covid19”, e em relação à organização de dados, os mesmos foram distribuídos por três discos (visto que não foram utilizadas máquinas virtuais, fizemos então a simulação de três discos em que cada um terá a sua própria pasta), para além do Primary Filegroup, foram criados também mais cinco Filegroup’s diferentes, em que em três deles estão incluídas as tabelas criadas, um deles estão incluídos os Indexes e no ultimo Filegroup é onde estão as logs da base de dados.

Como podemos ver na figura abaixo, o Filegroup primário está dividido em dois ficheiros:

```
CREATE DATABASE COVID19
ON PRIMARY
(
    NAME = Covid19_Data_Primary,
    FILENAME = 'D:\Escola\Projeto_BD2\Disco_C\Covid19_Data_Primary.mdf',
    SIZE = 10MB,
    MAXSIZE = 10GB,
    FILEGROWTH = 15%
),
(
    NAME = Covid19_Data_Primary_2,
    FILENAME = 'D:\Escola\Projeto_BD2\Disco_D\Covid19_Data_Primary_2.ndf',
    SIZE = 10MB,
    MAXSIZE = 10GB,
    FILEGROWTH = 15%
),
```

Figura 8 - Primary Filegroup

As figuras seguintes mostram a criação dos Filegroups das tabelas:

```
FILEGROUP Covid_Filegroup_Tables
(
    NAME = Covid_Data_Tables,
    FILENAME = 'D:\Escola\Projeto_BD2\Disco_C\Covid19_Data_Tables.ndf',
    SIZE = 5MB,
    MAXSIZE = 5GB,
    FILEGROWTH = 10%
),
(
    NAME = Covid19_Data_Tables_2,
    FILENAME = 'D:\Escola\Projeto_BD2\Disco_C\Covid19_Data_Tables_2.ndf',
    SIZE = 5MB,
    MAXSIZE = 5GB,
    FILEGROWTH = 10%
),
```

Figura 9 - Covid_Filegroup_Tables

Criação da Base de Dados e Gestão dos seus Elementos (Continuação)

```
FILEGROUP Covid19_Filegroup_Tables_2
(
    NAME = Covid19_Data_Tables_3,
    FILENAME = 'D:\Escola\Projeto_BD2\Disco_C\Covid19_Data_Tables_3.ndf',
    SIZE = 5MB,
    MAXSIZE = 5GB,
    FILEGROWTH = 10%
),
(
    NAME = Covid19_Data_Tables_4,
    FILENAME = 'D:\Escola\Projeto_BD2\Disco_D\Covid19_Data_Tables_4.ndf',
    SIZE = 5MB,
    MAXSIZE = 5GB,
    FILEGROWTH = 10%
),
```

Figura 10 – Covid19_Filegroup_Tables_2

```
FILEGROUP Covid19_Filegroup_Tables_3
(
    NAME = Covid19_Data_Tables_5,
    FILENAME = 'D:\Escola\Projeto_BD2\Disco_D\Covid19_Data_Tables_5.ndf',
    SIZE = 5MB,
    MAXSIZE = 5GB,
    FILEGROWTH = 10%
),
(
    NAME = Covid19_Data_Tables_6,
    FILENAME = 'D:\Escola\Projeto_BD2\Disco_D\Covid19_Tables_6.ndf',
    SIZE = 5MB,
    MAXSIZE = 5GB,
    FILEGROWTH = 10%
),
```

Figura 11 - Covid19_Filegroup_Tables_3

Criação da Base de Dados e Gestão dos seus Elementos (Continuação 2)

A cada um dos Filegroup's abaixo adicionadas tabelas diferentes de acordo com a sua importância e utilidade para a base de dados.

Na figura seguinte podemos observar a criação do Filegroup relativo aos Indexes.

```
FILEGROUP Covid19_Filegroup_Index
(
    NAME = Covid19_Data_Indexes,
    FILENAME = 'D:\Escola\Projeto_BD2\Disco_E\Covid_Data_Indexes.ndf',
    SIZE = 3MB,
    MAXSIZE = 3GB,
    FILEGROWTH = 5%
)
```

Figura 12 - Covid19_Filegroup_Index

Por fim, na figura abaixo podemos observar qual vai ser a organização dos ficheiros log.

```
LOG ON
(
    NAME = Covid19_log,
    FILENAME = 'D:\Escola\Projeto_BD2\Disco_E\Covid19_log.ldf',
    SIZE = 3MB,
    MAXSIZE = 3GB,
    FILEGROWTH = 5%
),
(
    NAME = Covid19_log_2,
    FILENAME = 'D:\Escola\Projeto_BD2\Disco_E\Covid19_log_2.ldf',
    SIZE = 3MB,
    MAXSIZE = 3GB,
    FILEGROWTH = 5%
);
```

Figura 13 - Organização dos Logs

Criação das Tabelas

Quanto à criação das tabelas na base de dados, foram criadas oito tabelas, em que nas figuras abaixo é mostrado os códigos em SQL para a criação das mesmas, visto que o Diagrama já foi mostrado na **Figura 4**.

<pre>CREATE TABLE Países (cod_País SMALLINT PRIMARY KEY, País VARCHAR(50), Abreviatura CHAR(2), Linguagem CHAR(3))</pre>	<pre>CREATE TABLE Pessoas (cod_Pessoa SMALLINT PRIMARY KEY, cod_Estado SMALLINT not null, FOREIGN KEY (cod_Estado) REFERENCES Estado(cod_Estado) ON UPDATE CASCADE ON DELETE CASCADE, Nome VARCHAR(100), cod_Cidade SMALLINT not null, FOREIGN KEY (cod_Cidade) REFERENCES Cidades(cod_Cidade) ON UPDATE CASCADE ON DELETE CASCADE, Nr_testes SMALLINT, Sexo VARCHAR(10), Idade SMALLINT, cod_GE SMALLINT not null, FOREIGN KEY (cod_GE) REFERENCES Grupo_etario(cod_GE) ON UPDATE CASCADE ON DELETE CASCADE, Morada VARCHAR(50), Localidade VARCHAR(50), NIF VARCHAR(30), Telefone VARCHAR(30), Email VARCHAR(50))</pre>
<pre>CREATE TABLE Regioes (cod_Regiao SMALLINT PRIMARY KEY, Regiao VARCHAR(50), cod_País SMALLINT not null, FOREIGN KEY (cod_País) REFERENCES Países(cod_País) ON UPDATE CASCADE ON DELETE CASCADE)</pre>	
<pre>CREATE TABLE Concelhos (cod_Concelho SMALLINT PRIMARY KEY, Concelho VARCHAR(50), Abreviatura CHAR(3))</pre>	
<pre>CREATE TABLE Estado (cod_Estado SMALLINT PRIMARY KEY, Designacao VARCHAR(5))</pre>	
<pre>CREATE TABLE Grupo_etario (cod_GE SMALLINT PRIMARY KEY, GrupoEtario VARCHAR(20))</pre>	<pre>CREATE TABLE Cidades (cod_Cidade SMALLINT PRIMARY KEY, cod_Regiao SMALLINT not null, FOREIGN KEY (cod_Regiao) REFERENCES Regioes(cod_Regiao) ON UPDATE CASCADE ON DELETE CASCADE, cod_Concelho SMALLINT not null, FOREIGN KEY (cod_Concelho) REFERENCES Concelhos(cod_Concelho) ON UPDATE CASCADE ON DELETE CASCADE, Cidade VARCHAR(100),)</pre>
<pre>CREATE TABLE Dados24h (cod_Dados SMALLINT PRIMARY KEY, cod_País SMALLINT not null, FOREIGN KEY (cod_País) REFERENCES Países(cod_País) ON UPDATE CASCADE ON DELETE CASCADE,)</pre>	

Figura 14 - Criação das Tabelas

Preenchimento da Base de Dados

Para o preenchimento da base de dados, houve parâmetros que foram inseridos manualmente, pois estes eram poucos e simples, como podemos ver na figura abaixo.

INSERT INTO Grupo_etario VALUES (1, '00-09'), (2, '10-19'), (3, '20-29'), (4, '30-39'), (5, '40-49'), (6, '50-59'), (7, '60-69'), (8, '70-79'), (9, '80+')	INSERT INTO Estado VALUES (1, 'Ativo'), (2, 'Recuperado'), (3, 'Óbito') INSERT INTO Dados24h VALUES (1, 0, 0, 0, 0, 0) INSERT INTO DadosTotal VALUES (1, 0, 0, 0, 0, 0)
--	---

Figura 15 - Preenchimento de Tabelas simples

Após a inserção desses dados, foram inseridos então alguns dados com apesar de terem sido um pouco mais complexos, foram inseridos através da utilização do programa Notepad++ e feita uma breve procura na Internet para que fossem inseridos os respetivos Concelhos e as suas respetivos Cidades, apesar destes dados terem sido inseridos manualmente, recordamos que estes parâmetros foram apenas feitos para o país de Portugal.

INSERT INTO Concelhos VALUES (1, 'Abrantes', 'ABT'), (2, 'Águeda', 'AGD'), (3, 'Aguiar da Beira', 'AGB'), (4, 'Alandroal', 'ADL'), (5, 'Albergaria-a-Velha', 'ALB'), (6, 'Albufeira', 'ABF'), (7, 'Alcácer do Sal', 'ASL'), (8, 'Alcanena', 'ACN'), (9, 'Alcobaça', 'ACB'), (10, 'Alcochete', 'ACH'),	INSERT INTO Cidades VALUES (1, 2519, 1, 'Aldeia do Mato e Souto'), (2, 2519, 1, 'Alvega e Concavada'), (3, 2519, 1, 'Bemposta'), (4, 2519, 1, 'Carvalhal'), (5, 2519, 1, 'Fontes'), (6, 2519, 1, 'Martinchel'), (7, 2519, 1, 'Mouriscas'), (8, 2519, 1, 'Pego'), (9, 2519, 1, 'Rio de Moinhos'), (10, 2519, 1, 'São Facundo e Vale das Mós'),
--	--

Figura 16 - Exemplo de Preenchimento de Tabelas Concelhos e Cidades

Como podemos ver na **Figura 15**, é mostrado um exemplo da inserção de Dados em ambas as tabelas, apenas são mostradas a inserção de 10 linhas visto que a tabela “Concelhos” contém **306 registos**(todos os concelhos de Portugal) e a tabela “Cidades” **3151 registos**(3047 são de Portugal e as restantes de outros países).

Preenchimentos da Base de Dados (Continuação)

Por fim, quanto à inserção de dados nas tabelas, foram inseridos os dados nas tabelas “Países”, “Regiões” e “Pessoas”, em que nessas tabelas o procedimento de inserção de dados já foi um pouco mais complexo, em que como serão inseridos milhares de dados o procedimento que usámos foi automático.

Por exemplo, no caso da tabela “Pessoas” foi usado uma ferramenta disponível online em que gera scripts de inserção de dados, essa ferramenta encontra-se disponível no website **Mockaroo**.

The screenshot shows the Mockaroo website interface for generating test data. The header is green with the Mockaroo logo and the text "realistic data generator". Below the header, there is a light gray box with introductory text and links. The main area is a table with three columns: "Field Name", "Type", and "Options". The table lists 12 fields for a table named "Pessoas". Each field has a "blank" checkbox, a percentage input (set to 0), and a "fx" button. At the bottom, there are controls for the number of rows (set to 1000), the output format (set to SQL), the table name (set to "Pessoas"), and a checkbox for "include create table".

Field Name	Type	Options
cod_Pessoa	Row Number	blank: 0 % <input checked="" type="button" value="fx"/>
cod_Estado	DUNS Number	blank: 0 % <input checked="" type="button" value="fx"/>
Nome	Full Name	blank: 0 % <input checked="" type="button" value="fx"/>
cod_Cidade	DUNS Number	blank: 0 % <input checked="" type="button" value="fx"/>
Nr_testes	DUNS Number	blank: 0 % <input checked="" type="button" value="fx"/>
Sexo	Gender (abbrev)	blank: 0 % <input checked="" type="button" value="fx"/>
Idade	DUNS Number	blank: 0 % <input checked="" type="button" value="fx"/>
Cod_GE	DUNS Number	blank: 0 % <input checked="" type="button" value="fx"/>
Morada	Street Address	blank: 0 % <input checked="" type="button" value="fx"/>
NIF	DUNS Number	blank: 0 % <input checked="" type="button" value="fx"/>
Telefone	DUNS Number	blank: 0 % <input checked="" type="button" value="fx"/>
Email	Email Address	blank: 0 % <input checked="" type="button" value="fx"/>

Rows: Format: Table Name: ☐ include create table

Figura 17 - Ferramenta utilizada para gerar Dados para a tabela Pessoas

A tabela “Pessoas” tem no total **3000 registos**, em que 2000 desses registos pertencem a pessoas de Portugal e 1000 a outros países.

Preenchimento da Base de Dados (Continuação 2)

Quanto ao preenchimento da tabela “Países” foi feita uma busca na internet para que fosse achada uma lista em que tivesse todos os países em determinada ordem, feito isso foi copiada e colada a lista de todos os países, em que depois foi adaptada essa lista à nossa base de dados, em que acrescentamos depois então os campos para que correspondessem a “Cod_Pais”, “Pais”, “Abreviatura”, “Linguagem” e “Populacao”, podemos verificar a criação desses campos se voltarmos a ver a **Figura 13**, onde foi feita a criação da mesma tabela.

```
INSERT INTO Países
VALUES
(1, 'Andorra', 'AD', 'ca', NULL),
(2, 'United Arab Emirates', 'Ae', 'ar', NULL),
(3, 'Afghanistan', 'AF', 'fa', NULL),
(4, 'Antigua and Barbuda', 'AG', 'en', NULL),
(5, 'Anguilla', 'AI', 'en', NULL),
(6, 'Albania', 'AL', 'sq', NULL),
(7, 'Armenia', 'AM', 'hy', NULL),
(8, 'Netherlands Antilles', 'A', 'nl', NULL),
(9, 'Angola', 'AO', 'pt', 33482000),
(10, 'Argentina', 'AR', 'es', NULL),
(11, 'Austria', 'AT', 'de', NULL),
(12, 'Australia', 'AU', 'en', 25499000),
```

Figura 18 - Inserção de dados na tabela Países

Na figura acima podemos ver um exemplo de inserção de 12 dados na tabela “Países”, em que foram apenas selecionados 12 visto que a tabela se encontra com **230 registros** no total. O mesmo procedimento de inserção de dados da tabela “Países” foi então usado igualmente para a inserção de dados na tabela “Regiões”.

```
INSERT INTO Regiões
VALUES
(1, 'Sant Julia de Loria', 1),
(2, 'Andorra la Vella', 1),
(3, 'La Massana', 1),
(4, 'Ordino', 1),
(5, 'Canillo', 1),
(6, 'Encamp', 1),
(7, 'Escaldes-Engordany', 1),
(8, 'Fujairah', 2),
(9, 'Abu Dhabi', 2),
(10, 'Dubai', 2),
```

Figura 19 - Inserção de dados na tabela Regiões

Na figura acima podemos ver então um exemplo de inserção de 10 regiões na sua tabela, visto que a tabela “Regiões” conta com um total de **3369 registros**.

Stored Procedures

Ao longo do projeto foi feita a criação de Stored Procedures, estas que, têm como função a dada pesquisa sobre um específico parâmetro. Nesta secção será então mostrada todas as Stored Procedures que foram feitas ao longo do projeto.

```
CREATE PROCEDURE sp_DadosPortugal
AS
BEGIN
    DECLARE @AtivosPortugal INT
    DECLARE @RecuperadosPortugal INT
    DECLARE @ObitosPortugal INT
    DECLARE @ConfirmadosPortugal INT
    DECLARE @Total_testesPortugal INT

    SET @AtivosPortugal = (SELECT COUNT(cod_Estado) FROM Pessoas
    INNER JOIN Cidades ON Pessoas.cod_Cidade = Cidades.cod_Cidade INNER JOIN Regioes ON Cidades.cod_Regiao = Regioes.cod_Regiao
    INNER JOIN Paises ON Regioes.cod_Pais = Paises.cod_Pais WHERE cod_Estado = 1 AND Paises.cod_Pais = 170)
    SET @RecuperadosPortugal = (SELECT COUNT(cod_Estado) FROM Pessoas
    INNER JOIN Cidades ON Pessoas.cod_Cidade = Cidades.cod_Cidade INNER JOIN Regioes ON Cidades.cod_Regiao = Regioes.cod_Regiao
    INNER JOIN Paises ON Regioes.cod_Pais = Paises.cod_Pais WHERE cod_Estado = 2 AND Paises.cod_Pais = 170)
    SET @ObitosPortugal = (SELECT COUNT(cod_Estado) FROM Pessoas
    INNER JOIN Cidades ON Pessoas.cod_Cidade = Cidades.cod_Cidade INNER JOIN Regioes ON Cidades.cod_Regiao = Regioes.cod_Regiao
    INNER JOIN Paises ON Regioes.cod_Pais = Paises.cod_Pais WHERE cod_Estado = 3 AND Paises.cod_Pais = 170)
    SET @ConfirmadosPortugal = (SELECT COUNT(cod_Estado) FROM Pessoas
    INNER JOIN Cidades ON Pessoas.cod_Cidade = Cidades.cod_Cidade INNER JOIN Regioes ON Cidades.cod_Regiao = Regioes.cod_Regiao
    INNER JOIN Paises ON Regioes.cod_Pais = Paises.cod_Pais WHERE Paises.cod_Pais = 170)
    SET @Total_testesPortugal = (SELECT SUM(Nr_testes) FROM Pessoas
    INNER JOIN Cidades ON Pessoas.cod_Cidade = Cidades.cod_Cidade INNER JOIN Regioes ON Cidades.cod_Regiao = Regioes.cod_Regiao
    INNER JOIN Paises ON Regioes.cod_Pais = Paises.cod_Pais WHERE Paises.cod_Pais = 170)

    SELECT @AtivosPortugal AS 'Casos Ativos', @RecuperadosPortugal AS 'Recuperados', @ObitosPortugal AS 'Óbitos', @ConfirmadosPortugal AS 'Casos Confirmados',
    @Total_testesPortugal AS 'Total de Testes Realizados'
END
```

Figura 20 - sp_DadosPortugal

Esta Stored Procedure tem como função apresentar o número de casos de covid19 ativos, recuperados, número de óbitos, número total de casos confirmados e também o número total de teste realizados, isto tudo apenas para o país de Portugal.

A procedure é executada com essa mesma função que podemos verificar o seu devido funcionamento na figura abaixo.

Results		Messages			
	Casos Ativos	Recuperados	Óbitos	Casos Confirmados	Total de Testes Realizados
1	635	703	662	2000	6041

Figura 21 - Resultados sp_DadosPortugal

Stored Procedures (2)

Foi criada uma procedure para ser feita a consulta de quantas pessoas infetados por covid19 mundialmente divididas por as suas idades correspondentes ao grupo etário que corresponde à sua idade.

```
CREATE PROCEDURE sp_DadosGE
AS
BEGIN
    SELECT Grupo_etario.GrupoEtario, COUNT(Pessoas.cod_GE)
    AS 'Quantidade' FROM Pessoas INNER JOIN Grupo_etario ON Pessoas.cod_GE = Grupo_etario.cod_GE
    GROUP BY GrupoEtario
END
```

Figura 22 - sp_DadosGE

A procedure seguinte foi criada com a função de contar todas as pessoas infetadas por covid19 no seu respetivo concelho.

```
CREATE PROCEDURE sp_DadosConcelhos
AS
BEGIN
    SELECT Concelhos.Concelho, COUNT(Pessoas.cod_Estado) AS 'Infetados' FROM Pessoas INNER JOIN Cidades ON Pessoas.cod_Cidade = Cidades.cod_Cidade
    INNER JOIN Concelhos ON Cidades.cod_Concelho = Concelhos.cod_Concelho WHERE cod_Estado = 1 GROUP BY Concelho
END
```

Figura 23 - sp_DadosConcelhos

Após serem executadas as seguintes procedures recebem os seguintes resultados:

	GrupoEtario	Quantidade
1	00-09	347
2	10-19	314
3	20-29	321
4	30-39	312
5	40-49	323
6	50-59	342
7	60-69	323
8	70-79	372
9	80+	346

Figura 24 - Resultado sp_DadosGE

	Concelho	Infetados
4	Alandroal	2
5	Alcácer do Sal	1
6	Alcobaça	1
7	Alcochete	1
8	Alcoutim	2
9	Alenquer	4
10	Alfândega da Fé	3
11	Aljezur	1
12	Aljustrel	1
13	Almada	2
14	Almeida	4
15	Alter do Chão	2

Figura 25 - Resultado sp_DadosConcelhos

Stored Procedures (3)

Para a seguinte procedure tivemos como objetivo calcular o número de pessoas infectadas por covid19 assim como, o número de recuperados, óbitos e o total de casos confirmados, em que tudo isto era feito para um respetivo concelho.

```
CREATE PROCEDURE sp_Concelho @Concelho VARCHAR(50)
AS
BEGIN
    DECLARE @Ativos INT
    DECLARE @Recuperados INT
    DECLARE @Obitos INT
    DECLARE @Confirmados INT

    SET @Ativos = ( SELECT COUNT(Pessoas.cod_Estado) FROM Pessoas INNER JOIN Cidades ON Pessoas.cod_Cidade = Cidades.cod_Cidade
    INNER JOIN Concelhos ON Cidades.cod_Concelho = Concelhos.cod_Concelho WHERE Concelho = @Concelho AND cod_Estado = 1)
    SET @Recuperados = (SELECT COUNT(Pessoas.cod_Estado) FROM Pessoas INNER JOIN Cidades ON Pessoas.cod_Cidade = Cidades.cod_Cidade
    INNER JOIN Concelhos ON Cidades.cod_Concelho = Concelhos.cod_Concelho WHERE Concelho = @Concelho AND cod_Estado = 2)
    SET @Obitos = (SELECT COUNT(Pessoas.cod_Estado) FROM Pessoas INNER JOIN Cidades ON Pessoas.cod_Cidade = Cidades.cod_Cidade
    INNER JOIN Concelhos ON Cidades.cod_Concelho = Concelhos.cod_Concelho WHERE Concelho = @Concelho AND cod_Estado = 3)
    SET @Confirmados = (SELECT COUNT(Pessoas.cod_Estado) FROM Pessoas INNER JOIN Cidades ON Pessoas.cod_Cidade = Cidades.cod_Cidade
    INNER JOIN Concelhos ON Cidades.cod_Concelho = Concelhos.cod_Concelho WHERE Concelho = @Concelho)

    SELECT @Concelho AS 'Concelho', @Ativos AS 'Ativos', @Recuperados AS 'Recuperados', @Obitos AS 'Obitos', @Confirmados AS 'Confirmados'
END
```

Figura 26 - sp_Concelho

Para a sua devida execução da procedure temos de executar o código que usamos normalmente para executar a procedure mas com a exceção de que igualamos a variável “@Concelho” para o concelho que queremos que os dados sejam consultados.

Ou seja, um exemplo do código a implementar iria ser assim:

```
EXEC sp_Concelho @Concelho = Lisboa
```

Após a execução desse código conseguimos obter os seguintes resultados:

Results		Messages			
	Concelho	Ativos	Recuperados	Obitos	Confirmados
1	Lisboa	5	5	4	14

Figura 27 - Resultado sp_Concelho

Stored Procedures (4)

Para a criação da procedure “sp_ConcelhoCidades” tivemos como objetivo criar uma procedure que apresente o concelho e todas as suas respectivas cidades em que em cada uma das suas cidades desse respetivo concelho apresenta a consulta das pessoas infetadas dessa mesma cidade.

```
ALTER PROCEDURE sp_ConcelhoCidades @Concelho VARCHAR(50)
AS
BEGIN
    SELECT @Concelho AS 'Concelho', Cidades.Cidade, COUNT(cod_Estado) AS 'Ativos'
    FROM Pessoas
    INNER JOIN Cidades ON Pessoas.cod_Cidade = Cidades.cod_Cidade
    INNER JOIN Concelhos ON Cidades.cod_Concelho = Concelhos.cod_Concelho
    WHERE cod_Estado = 1 AND Concelhos.Concelho = @Concelho
    GROUP BY Cidades.Cidade
END
```

Figura 28 - sp_ConcelhoCidades

Assim como na procedure anterior (**Figura 26**), executamos a procedure da mesma maneira, em que executamos o código com o respetivo concelho para os dados que queremos consultar corresponderem às cidades com a quantidade de infetados pertencentes a esse concelho previamente inserido:

```
EXEC sp_ConcelhoCidades @Concelho = Lisboa
```

	Concelho	Cidade	Ativos
1	Lisboa	Alvalade	1
2	Lisboa	Arroios	1
3	Lisboa	Camide	2
4	Lisboa	Misericórdia	1

Figura 29 - Resultados sp_ConcelhoCidades

Stored Procedures (5)

Para a criação da seguinte procedure pretendemos criar uma consulta para saber todos os casos ativos, recuperados, óbitos e todos os casos confirmados em que todos são agrupados por seu respetivo sexo (Masculino e Feminino).

```
CREATE PROCEDURE sp_TotalGenero
AS
BEGIN
    SELECT Sexo, COUNT(CASE cod_Estado WHEN 1 THEN 1 ELSE NULL END) AS 'Ativos',
    COUNT(CASE cod_Estado WHEN 2 THEN 1 ELSE NULL END) AS 'Recuperados',
    COUNT(CASE cod_Estado WHEN 3 THEN 1 ELSE NULL END) AS 'Obitos',
    COUNT(cod_Estado) AS 'Confirmados'
    FROM Pessoas
    GROUP BY Sexo
END
```

Figura 30 - sp_TotalGenero

Logo depois dessa procedure foi feita uma para ser feita uma consulta por país em que mostra todas as regiões do país procurado e todos os casos ativos de covid nessas regiões do país em procura.

```
CREATE PROCEDURE sp_AtivosRegiao @Pais VARCHAR(50)
AS
BEGIN
    SELECT @Pais AS 'País', Regioes.Regiao, COUNT(cod_Estado) AS 'Ativos'
    FROM Pessoas
    INNER JOIN Cidades ON Pessoas.cod_Cidade = Cidades.cod_Cidade
    INNER JOIN Regioes ON Cidades.cod_Regiao = Regioes.cod_Regiao INNER JOIN Países ON Regioes.cod_Pais = Países.cod_Pais
    WHERE cod_Estado = 1 AND Países.Pais = @Pais
    GROUP BY Regioes.Regiao
END
```

Figura 31 - sp_AtivosRegiao

Nas figuras abaixo podemos ver os resultados da consulta de ambas as procedures, “sp_TotalGenero” e “sp_AtivosRegiao”:

	Sexo	Ativos	Recuperados	Obitos	Confirmados
1	F	482	514	520	1516
2	M	481	526	477	1484

Figura 32 - Resultados sp_TotalGenero

	País	Regiao	Ativos
1	Portugal	Açores	37
2	Portugal	Alentejo	66
3	Portugal	Algarve	15
4	Portugal	Centro	181
5	Portugal	Lisboa e Vale do Tejo	28
6	Portugal	Madeira	8
7	Portugal	Norte	300

Figura 33 - Resultados sp_AtivosRegiao

Stored Procedures (6)

As seguintes 3 procedures foram criadas como variações da procedure “sp_AtivosRegiao” (**Figura 31**), em que foram criadas 3 procedures para obter três resultados diferentes que vairam consoante uns dos outros, seguem as figuras.

```
CREATE PROCEDURE sp_RecuperadosRegiao @Pais VARCHAR(50)
AS
BEGIN
    SELECT @Pais AS 'País', Regioes.Regiao, COUNT(cod_Estado) AS 'Recuperados'
    FROM Pessoas
    INNER JOIN Cidades ON Pessoas.cod_Cidade = Cidades.cod_Cidade
    INNER JOIN Regioes ON Cidades.cod_Regiao = Regioes.cod_Regiao INNER JOIN Paises ON Regioes.cod_Pais = Paises.cod_Pais
    WHERE cod_Estado = 2 AND Paises.Pais = @Pais
    GROUP BY Regioes.Regiao
END
```

Figura 34 - sp_RecuperadosRegiao

```
CREATE PROCEDURE sp_ObitosRegiao @Pais VARCHAR(50)
AS
BEGIN
    SELECT @Pais AS 'País', Regioes.Regiao, COUNT(cod_Estado) AS 'Obitos'
    FROM Pessoas
    INNER JOIN Cidades ON Pessoas.cod_Cidade = Cidades.cod_Cidade
    INNER JOIN Regioes ON Cidades.cod_Regiao = Regioes.cod_Regiao INNER JOIN Paises ON Regioes.cod_Pais = Paises.cod_Pais
    WHERE cod_Estado = 3 AND Paises.Pais = @Pais
    GROUP BY Regioes.Regiao
END
```

Figura 35 - sp_ObitosRegiao

```
CREATE PROCEDURE sp_ConfirmadosRegiao @Pais VARCHAR(50)
AS
BEGIN
    SELECT @Pais AS 'País', Regioes.Regiao, COUNT(cod_Estado) AS 'Confirmados'
    FROM Pessoas
    INNER JOIN Cidades ON Pessoas.cod_Cidade = Cidades.cod_Cidade
    INNER JOIN Regioes ON Cidades.cod_Regiao = Regioes.cod_Regiao INNER JOIN Paises ON Regioes.cod_Pais = Paises.cod_Pais
    WHERE Paises.Pais = @Pais
    GROUP BY Regioes.Regiao
END
```

Figura 36 - sp_ConfirmadosRegiao

Stored Procedures (7)

Procedure para verificar a quantidade existente de óbitos por grupo etário, em que representa os valores dos óbitos por covid em seguimento dos seus grupos de idade.

```
CREATE PROCEDURE sp_MortesPorIdade
AS
BEGIN
    SELECT Grupo_etario.GrupoEtario, COUNT(Pessoas.cod_GE)
    AS 'Óbitos' FROM Pessoas INNER JOIN Grupo_etario ON Pessoas.cod_GE = Grupo_etario.cod_GE
    WHERE cod_Estado = 3
    GROUP BY GrupoEtario
END
```

Figura 37 - sp_MortesPorIdade

Após isso foi feita uma procedure em que verifica o número total de casos ativos de pessoas em cada país

```
CREATE PROCEDURE sp_AtivosPorPais
AS
BEGIN
    SELECT Pais.Pais, COUNT(Pessoas.cod_Estado) AS 'Ativos' FROM Pessoas INNER JOIN Cidades ON Pessoas.cod_Cidade = Cidades.cod_Cidade
    INNER JOIN Regioes ON Cidades.cod_Regiao = Regioes.cod_Regiao INNER JOIN Pais ON Regioes.cod_Pais = Pais.cod_Pais WHERE cod_Estado =1 GROUP BY Pais
END
```

Figura 38 – sp_AtivosPorPais

Abaixo seguem-se os resultados das procedures “sp_MortesPorIdade” e “sp_AtivosPorPais”

	GrupoEtario	Óbitos
1	00-09	117
2	10-19	120
3	20-29	88
4	30-39	103
5	40-49	110
6	50-59	113
7	60-69	104
8	70-79	126
9	80+	116

Figura 39 - Resultados "sp_MortesPorIdade"

	Pais	Ativos
1	Alemanha	11
2	Angola	21
3	Australia	14
4	Bélgica	17
5	Brasil	25
6	China	20
7	Espanha	95
8	Estados Unidos da América	26
9	França	21
10	Holanda	12
11	Itália	19
12	Japão	21

Figura 40 - Resultados "sp_AtivosPorPais"

Stored Procedures (8)

As seguintes 3 procedures foram criadas como variações da procedure “sp_AtivosPorPais” (**Figura 38**), em que foram criadas 3 procedures para obter três resultados diferentes que vairam consoante uns dos outros, seguem as figuras.

```
CREATE PROCEDURE sp_RecuperadosPorPais
AS
BEGIN
    SELECT Pais.Pais, COUNT(Pessoas.cod_Estado) AS 'Recuperados' FROM Pessoas INNER JOIN Cidades ON Pessoas.cod_Cidade = Cidades.cod_Cidade
    INNER JOIN Regioes ON Cidades.cod_Regiao = Regioes.cod_Regiao INNER JOIN Pais ON Regioes.cod_Pais = Pais.cod_Pais WHERE cod_Estado = 2 GROUP BY Pais
END
```

Figura 41 - sp_RecuperadosPorPais

```
CREATE PROCEDURE sp_ObitosPorPais
AS
BEGIN
    SELECT Pais.Pais, COUNT(Pessoas.cod_Estado) AS 'Obitos' FROM Pessoas INNER JOIN Cidades ON Pessoas.cod_Cidade = Cidades.cod_Cidade
    INNER JOIN Regioes ON Cidades.cod_Regiao = Regioes.cod_Regiao INNER JOIN Pais ON Regioes.cod_Pais = Pais.cod_Pais WHERE cod_Estado = 3 GROUP BY Pais
END
```

Figura 42 - sp_ObitosPorPais

```
CREATE PROCEDURE sp_ConfirmadosPorPais
AS
BEGIN
    SELECT Pais.Pais, COUNT(Pessoas.cod_Estado) AS 'Confirmados' FROM Pessoas INNER JOIN Cidades ON Pessoas.cod_Cidade = Cidades.cod_Cidade
    INNER JOIN Regioes ON Cidades.cod_Regiao = Regioes.cod_Regiao INNER JOIN Pais ON Regioes.cod_Pais = Pais.cod_Pais GROUP BY Pais
END
```

Figura 43 - sp_ConfirmadosPorPais

Triggers

Assim como as stored procedures, ao longo do projeto foi também feita a criação de Triggers, estes que, têm como função serem executados assim que uma dada ação acontece, ou seja, “disparam” a dada ação.

Nas figuras abaixo são exemplificados alguns exemplos de triggers que foram utilizados no projeto.

```
CREATE TRIGGER tr_Dados24h
ON dbo.Pessoas
AFTER UPDATE, INSERT
AS
    DECLARE @DataAnterior DATETIME
    SET @DataAnterior = DATEADD(HOUR,-24,SYSDATETIME())
    UPDATE dbo.Pessoas
    SET Data_mod = SYSDATETIME()
    FROM Inserted i
    WHERE dbo.Pessoas.cod_Pessoa = i.cod_Pessoa
    UPDATE dbo.Dados24h
    SET Ativos = (SELECT COUNT(cod_Estado) FROM Pessoas
    WHERE cod_Estado = 1 AND Data_mod BETWEEN @DataAnterior AND SYSDATETIME()),
    Recuperados = (SELECT COUNT(cod_Estado) FROM Pessoas
    WHERE cod_Estado = 2 AND Data_mod BETWEEN @DataAnterior AND SYSDATETIME()),
    Obitos = (SELECT COUNT(cod_Estado) FROM Pessoas
    WHERE cod_Estado = 3 AND Data_mod BETWEEN @DataAnterior AND SYSDATETIME()),
    Confirmados = (SELECT COUNT(cod_Estado) FROM Pessoas
    WHERE Data_mod BETWEEN @DataAnterior AND SYSDATETIME()),
    Total_testes = (SELECT SUM(Nr_testes) FROM Pessoas
    WHERE Data_mod BETWEEN @DataAnterior AND SYSDATETIME())
```

Figura 44 - tr_Dados24h

Este trigger “tr_Dados24” faz com que ao alterar ou inserir algum campo na tabela “Pessoas” irá gerar automaticamente a data atual do sistema quando foi feita essa alteração ou adição e irá então colocar essa data que foi gerada na coluna “Data_mod” da tabela “Pessoas”, e que com isso irá alterar os valores da tabela “Dados24h” que irão gerar os dados mundiais em relação covid dentro de 24 horas a partir disso.

Results		Messages				
	cod_Dados	Ativos	Recuperados	Obitos	Confirmados	Total_testes
1	1	963	1040	997	3000	9007

Figura 45 - Tabela "Dados24h"

Triggers (2)

O seguinte trigger é um pouco semelhante ao anterior segue o mesmo procedimento, o trigger irá “disparar” assim que algum campo seja alterado ou inserido na tabela “Pessoas”, só que neste trigger ao contrário do anterior irá armazenar sempre os valores dos casos ativos, recuperados, óbitos, confirmados e total de testes, em que no caso do trigger anterior ao fim de 24h os dados começam do zero.

```
CREATE TRIGGER tr_DadosTotal
ON dbo.Pessoas
AFTER UPDATE, INSERT
AS
    UPDATE dbo.DadosTotal
    SET Ativos = (SELECT COUNT(cod_Estado) FROM Pessoas
    WHERE cod_Estado = 1),
    Recuperados = (SELECT COUNT(cod_Estado) FROM Pessoas
    WHERE cod_Estado = 2),
    Obitos = (SELECT COUNT(cod_Estado) FROM Pessoas
    WHERE cod_Estado = 3),
    Confirmados = (SELECT COUNT(cod_Estado) FROM Pessoas),
    Total_testes = (SELECT SUM(Nr_testes) FROM Pessoas)
```

Figura 46 - tr_DadosTotal

Results		Messages				
	cod_Dados	Ativos	Recuperados	Obitos	Confirmados	Total_testes
1	1	963	1040	997	3000	9007

Figura 47 - Tabela "DadosTotal"

O trigger na figura abaixo (“tr_Seguranca”), serve para prevenir que alguma coluna da tabela seja alterada ou até mesmo apagada, em que se caso isso aconteça o utilizador receberá a mensagem “Não pode apagar ou alterar tabelas” e irá acontecer um Rollback em que a tabela irá ser restaurada aos valores iniciais antes da sua alteração.

```
CREATE TRIGGER tr_Seguranca
ON DATABASE
FOR DROP_TABLE, ALTER_TABLE
AS
BEGIN
    PRINT 'Não pode apagar ou alterar tabelas'
    ROLLBACK
END
```

Figura 48 - tr_Seguranca

Triggers (3)

O seguinte trigger serve para a partir do momento que é inserido ou alterado um campo na tabela “Pessoas” este trigger irá verificar se o número da população infetada de determinado país é igual ou excede o 1% da população total, em que caso isso acontece, ele irá mostrar um alerta.

```
CREATE TRIGGER tr_AlertaPopulacao
ON dbo.Pessoas
AFTER UPDATE, INSERT
AS
    DECLARE @PopulacaoPortugal DECIMAL(15,5)
    DECLARE @PopulacaoEspanha DECIMAL(15,5)
    DECLARE @PopulacaoTESTE DECIMAL(15,10)
    DECLARE @AtivosPortugal DECIMAL(15,10)
    DECLARE @AtivosEspanha DECIMAL(15,10)
    DECLARE @AtivosTESTE DECIMAL(15,10)

    SET @PopulacaoPortugal = (SELECT Populacao FROM Países WHERE cod_Pais = 170)
    SET @PopulacaoEspanha = (SELECT Populacao FROM Países WHERE cod_Pais = 62)
    SET @PopulacaoTESTE = (SELECT Populacao FROM Países WHERE cod_Pais = 231)

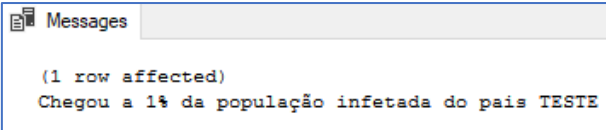
    SET @AtivosPortugal = (SELECT COUNT(cod_Estado) FROM Pessoas
    INNER JOIN Cidades ON Pessoas.cod_Cidade = Cidades.cod_Cidade INNER JOIN Regioes ON Cidades.cod_Regiao = Regioes.cod_Regiao
    INNER JOIN Países ON Regioes.cod_Pais = Países.cod_Pais WHERE cod_Estado = 1 AND Países.cod_Pais = 170)
    SET @AtivosEspanha = (SELECT COUNT(cod_Estado) FROM Pessoas
    INNER JOIN Cidades ON Pessoas.cod_Cidade = Cidades.cod_Cidade INNER JOIN Regioes ON Cidades.cod_Regiao = Regioes.cod_Regiao
    INNER JOIN Países ON Regioes.cod_Pais = Países.cod_Pais WHERE cod_Estado = 1 AND Países.cod_Pais = 62)
    SET @AtivosTESTE = (SELECT COUNT(cod_Estado) FROM Pessoas
    INNER JOIN Cidades ON Pessoas.cod_Cidade = Cidades.cod_Cidade INNER JOIN Regioes ON Cidades.cod_Regiao = Regioes.cod_Regiao
    INNER JOIN Países ON Regioes.cod_Pais = Países.cod_Pais WHERE cod_Estado = 1 AND Países.cod_Pais = 231)

    IF @AtivosPortugal / @PopulacaoPortugal * 100 >= 1
    BEGIN
        PRINT 'Chegou a 1% da população infetada de Portugal'
    END

    IF @AtivosEspanha / @PopulacaoEspanha * 100 >= 1
    BEGIN
        PRINT 'Chegou a 1% da população infetada de Espanha'
    END

    IF @AtivosTESTE / @PopulacaoTESTE * 100 >= 1
    BEGIN
        PRINT 'Chegou a 1% da população infetada do país TESTE'
    END
```

Figura 49 - tr_AlertaPopulacao



Messages

(1 row affected)
Chegou a 1% da população infetada do país TESTE

Figura 50 - Resultado "tr_AlertaPopulacao"

Medidas de Segurança

Em termos de medidas de segurança foram criadas “roles” e “users” de maneira a atribuir permissões de execução a certos stored procedures.

Quanto aos “roles”, foram criados três. Um “role” de administração (**db_Administrador**), um “role” para os concelhos (**db_Concelho**) e um “role” de visitante (**db_Visitante**).

```
EXEC sp_addrole 'db_Administrador'  
GO  
EXEC sp_addrole 'db_Concelho'  
GO  
EXEC sp_addrole 'db_Visitante'
```

Figura 51 – Roles

Foram também criados dois “users” para que esses possam usufruir das “roles”.

```
CREATE USER Sines WITHOUT LOGIN  
GO  
CREATE USER DGS WITHOUT LOGIN  
GO  
  
EXEC sp_addrolemember 'db_owner', 'db_Administrador'  
GO  
EXEC sp_addrolemember 'db_Concelho', 'Sines'  
GO  
EXEC sp_addrolemember 'db_Visitante', 'DGS'
```

Figura 52 - Users

Aos “roles” concelhos (**db_Concelho**) e visitante (**db_Visitante**) foram dadas permissões de execução para alguns stored procedures de modo a conseguirem visualizar a informação.

<pre>GRANT EXEC ON sp_Concelho to db_Concelho GO GRANT EXEC ON sp_ConcelhoCidades to db_Concelho GO GRANT EXEC ON sp_DadosPortugal to db_Visitante GO GRANT EXEC ON sp_DadosConcelhos to db_Visitante GO GRANT EXEC ON sp_Concelho to db_Visitante GO GRANT EXEC ON sp_ConcelhoCidades to db_Visitante GO GRANT EXEC ON sp_DadosGE to db_Visitante GO GRANT EXEC ON sp_TotalGenero to db_Visitante</pre>	<pre>GRANT EXEC ON sp_AtivosRegiao to db_Visitante GO GRANT EXEC ON sp_RecuperadosRegiao to db_Visitante GO GRANT EXEC ON sp_ObitosRegiao to db_Visitante GO GRANT EXEC on sp_ConfirmadosRegiao to db_Visitante GO GRANT EXEC ON sp_AtivosPorPais to db_Visitante GO GRANT EXEC ON sp_RecuperadosPorPais to db_Visitante GO GRANT EXEC ON sp_ObitosPorPais to db_Visitante GO GRANT EXEC ON sp_ConfirmadosPorPais to db_Visitante</pre>
--	---

Figura 53 - Permissões

Cópias de Segurança

Neste ponto serão especificados os tipos de cópias de segurança, e a sua prioridade bem como a maneira de como foram criados.

Por se tratar de uma base de dados que irá ter milhares de acessos todos os dias, decidiu-se que a política de cópias de segurança a tomar seria a seguinte:

- **Full Backups:** dois por dia, um às 00:00h e outro às 12:00h;
- **Differential Backups:** de 4 em 4 horas;
- **Logs:** backups de 15 em 15 minutos.

Para garantir a periodicidade dos backups criou-se um **Maintaince Plan**, como se pode ver nas seguintes figuras:

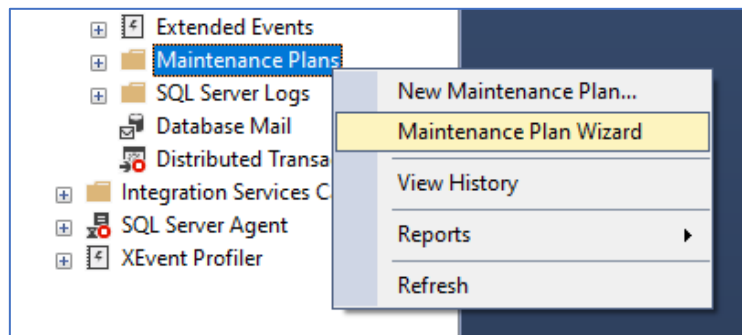


Figura 54 - Passo 1

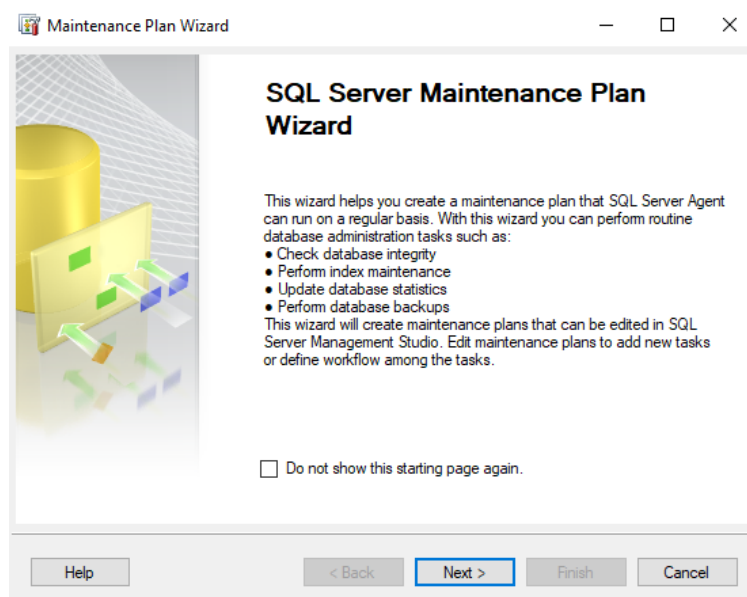


Figura 55 - Passo 2

Cópias de Segurança (2)

Maintenance Plan Wizard

Select Plan Properties
How do you want to schedule your maintenance tasks?

Name: COVID19_Backup_MaintenancePlan

Description:

Run as: SQL Server Agent service account

☒ Separate schedules for each task
☐ Single schedule for the entire plan or no schedule

Schedule: Not scheduled (On Demand) Change...

Help < Back **Next >** Finish Cancel

Figura 56 - Passo 3

Maintenance Plan Wizard

Select Maintenance Tasks
Which tasks should this plan perform?

Select one or more maintenance tasks:

- ☒ Check Database Integrity
- ☐ Shrink Database
- ☐ Reorganize Index
- ☐ Rebuild Index
- ☐ Update Statistics
- ☐ Clean Up History
- ☐ Execute SQL Server Agent Job
- ☒ Back Up Database (Full)
- ☒ Back Up Database (Differential)
- ☒ Back Up Database (Transaction Log)
- ☐ Maintenance Cleanup Task

i The Back Up Database (Full) task allows you to specify the source databases, destination files or tapes, and overwrite options for a full backup.

Help < Back **Next >** Finish Cancel

Figura 57 - Passo 4

Cópias de Segurança (3)

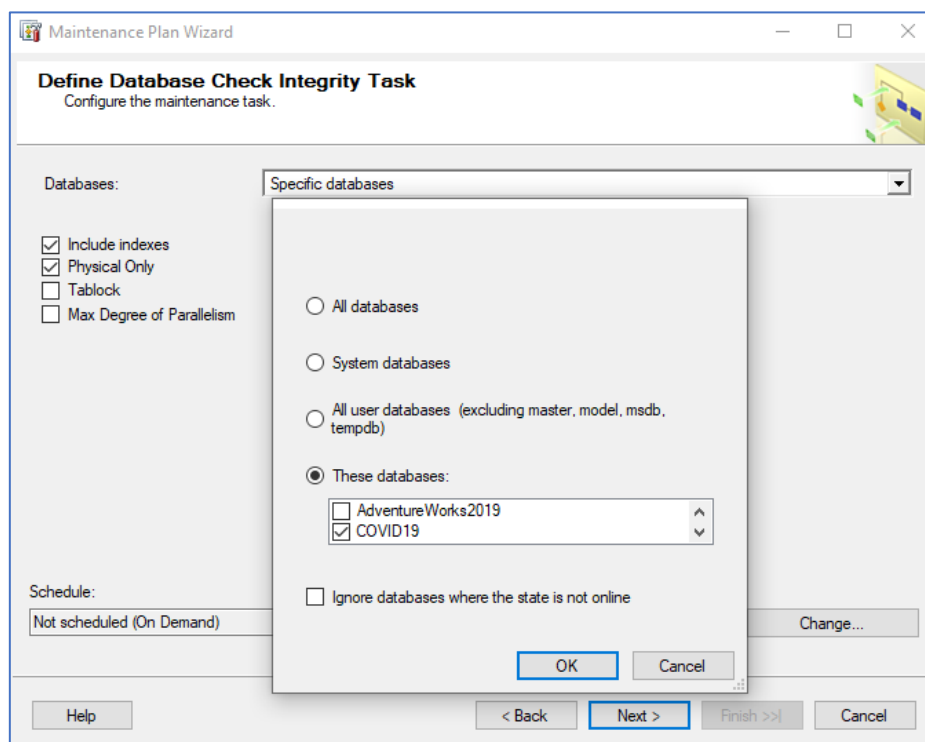


Figura 58 - Passo 5

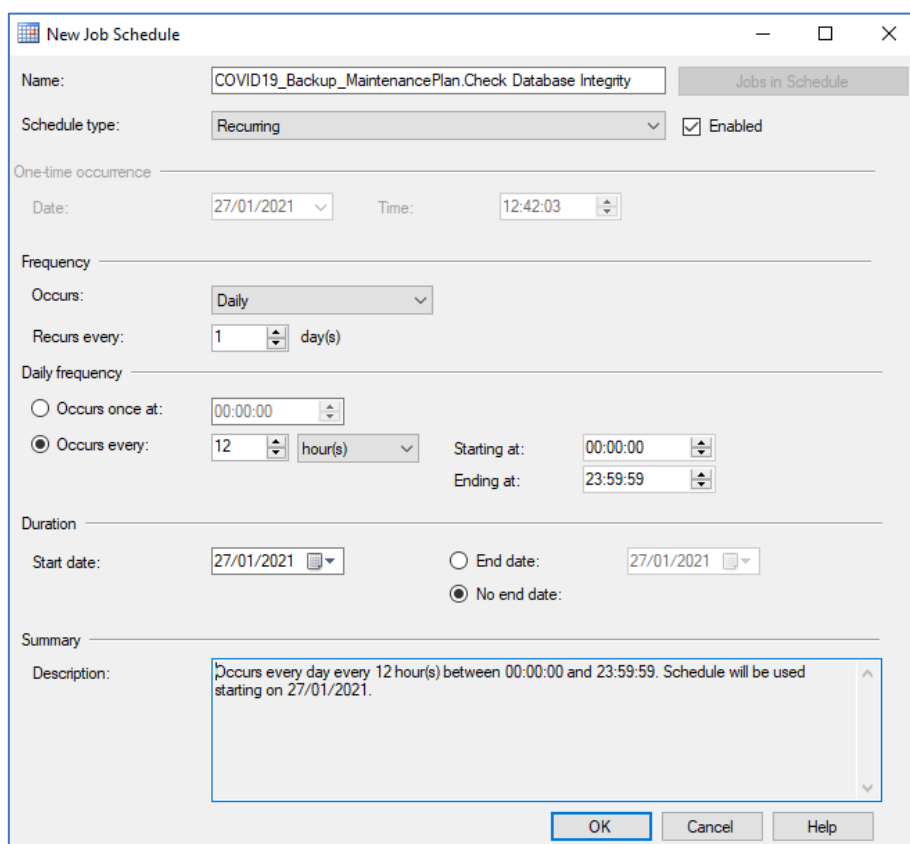


Figura 59 - Passo 6

Cópias de Segurança (4)

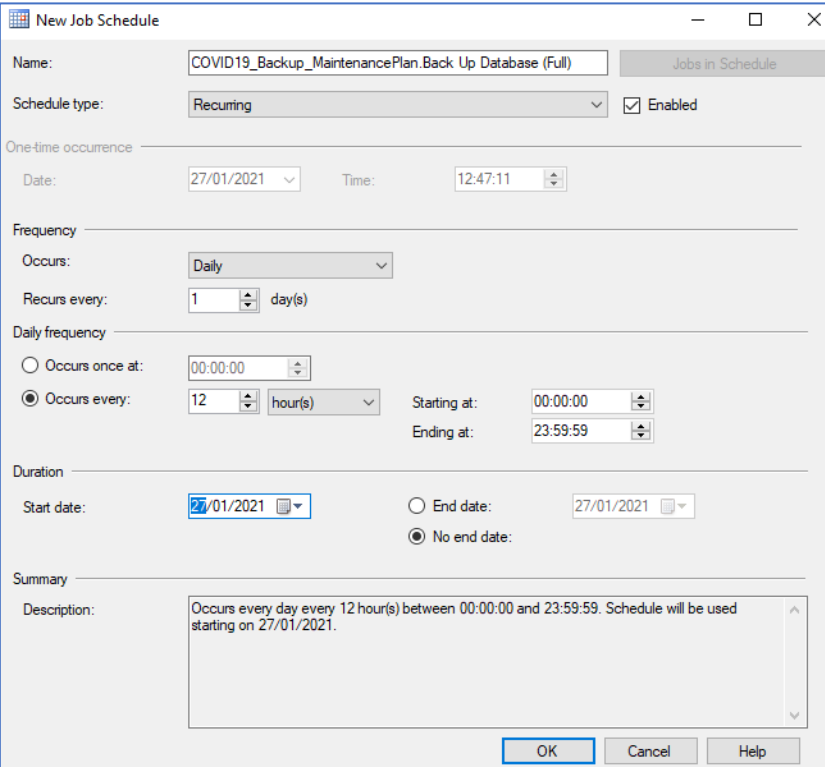
The screenshot shows the 'Maintenance Plan Wizard' window at the 'Define Database Check Integrity Task' step. The title bar reads 'Maintenance Plan Wizard'. The main heading is 'Define Database Check Integrity Task' with the subtitle 'Configure the maintenance task.' Below this, there is a 'Databases:' dropdown menu set to 'Specific databases'. There are four checkboxes: 'Include indexes' (checked), 'Physical Only' (checked), 'Tablock' (unchecked), and 'Max Degree of Parallelism' (unchecked) with a value of '1' in a spinner box. At the bottom, the 'Schedule:' section shows 'Occurs every day every 12 hour(s) between 00:00:00 and 23:59:59. Schedule will be used starting on 2' with a 'Change...' button. Navigation buttons at the bottom include 'Help', '< Back', 'Next >', 'Finish >>', and 'Cancel'.

Figura 60 - Passo 7

The screenshot shows the 'Maintenance Plan Wizard' window at the 'Define Back Up Database (Full) Task' step. The title bar reads 'Maintenance Plan Wizard'. The main heading is 'Define Back Up Database (Full) Task' with the subtitle 'Configure the maintenance task.' The 'General' tab is selected. 'Backup type:' is set to 'Full'. 'Database(s):' is set to '<Select one or more>'. 'Backup component' has 'Database' selected. 'Back up to:' is set to 'Disk'. A dialog box is open for selecting databases, showing options: 'All databases', 'System databases', 'All user databases (excluding master, model, msdb, tempdb)', and 'These databases:'. Under 'These databases:', a list includes 'AdventureWorks2019', 'COVID19' (checked), 'Marketing', 'master', and 'model'. There is also an option 'Ignore databases where the state is not online' (unchecked). The dialog has 'OK' and 'Cancel' buttons. The background wizard shows a 'Schedule:' of 'Not scheduled (On Demand)' and navigation buttons at the bottom.

Figura 61 - Passo 8

Cópias de Segurança (5)

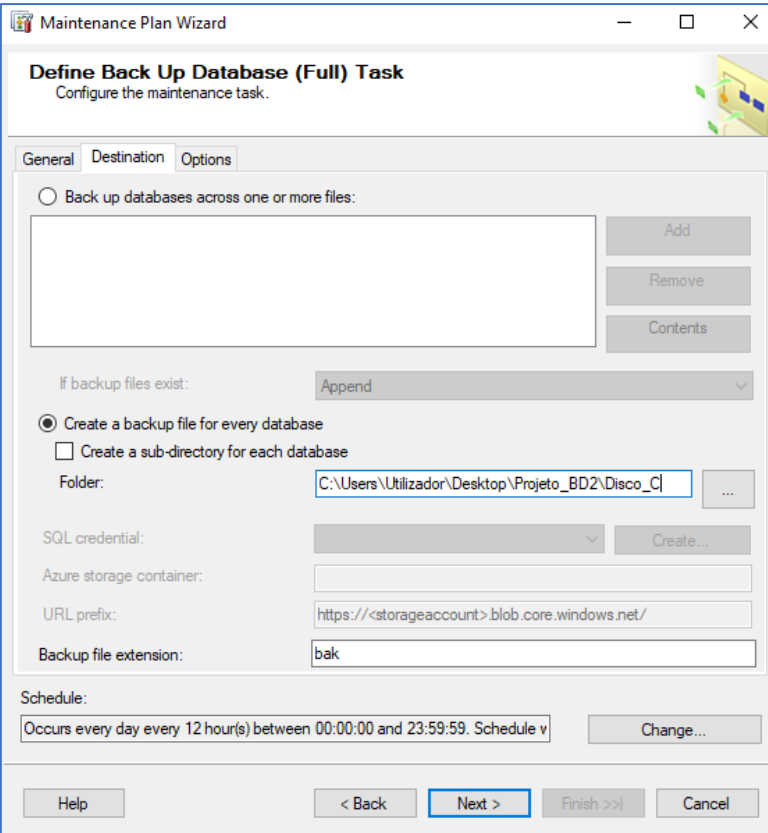


The 'New Job Schedule' dialog box is shown with the following configuration:

- Name:** COVID19_Backup_MaintenancePlan.Back Up Database (Full)
- Schedule type:** Recurring
- Enabled:** ☒
- One-time occurrence:** Date: 27/01/2021, Time: 12:47:11
- Frequency:**
 - Occurs:** Daily
 - Recurs every:** 1 day(s)
- Daily frequency:**
 - ☐ Occurs once at: 00:00:00
 - ☒ Occurs every: 12 hour(s), Starting at: 00:00:00, Ending at: 23:59:59
- Duration:**
 - Start date:** 27/01/2021
 - ☐ End date: 27/01/2021
 - ☒ No end date:
- Summary:**
 - Description:** Occurs every day every 12 hour(s) between 00:00:00 and 23:59:59. Schedule will be used starting on 27/01/2021.

Buttons: OK, Cancel, Help

Figura 62 - Passo 9



The 'Maintenance Plan Wizard' is shown at the 'Define Back Up Database (Full) Task' step. The 'General' tab is selected.

Define Back Up Database (Full) Task
Configure the maintenance task.

General | Destination | Options

- ☐ Back up databases across one or more files:
 - Buttons: Add, Remove, Contents
- ☒ Create a backup file for every database
 - ☐ Create a sub-directory for each database
 - Folder:** C:\Users\Utilizador\Desktop\Projeto_BD2\Disco_C | ...
 - SQL credential:** [Dropdown] | Create...
 - Azure storage container:** [Text box]
 - URL prefix:** https://<storageaccount>.blob.core.windows.net/
 - Backup file extension:** bak

Schedule: Occurs every day every 12 hour(s) between 00:00:00 and 23:59:59. Schedule v | Change...

Buttons: Help, < Back, Next >, Finish >>, Cancel

Figura 63 - Passo 10

Cópias de Segurança (6)

Define Back Up Database (Full) Task
Configure the maintenance task.

General Destination Options

Set backup compression: Use the default server setting

☐ Backup set will expire:

☒ After 14 days

☐ On 10/02/2021

☐ Copy-only backup

☒ Verify backup integrity

☒ Perform checksum

☐ Backup encryption

Algorithm: AES 128

Certificate or Asymmetric key:

☐ For availability databases, ignore replica priority for backup and backup on primary settings

☐ Block size 65536 bytes

☐ Max transfer size 65536 bytes

Schedule:
Occurs every day every 12 hour(s) between 00:00:00 and 23:59:59. Schedule v

Help < Back Next > Finish >> Cancel

Figura 64 - Passo 11

Define Back Up Database (Differential) Task
Configure the maintenance task.

General Destination Options

☐ Back up databases across one or more files:

If backup files exist: Append

☒ Create a backup file for every database

☐ Create a sub-directory for each database

Folder: C:\Users\Utilizador\Desktop\Projeto_BD2\Disco_D

SQL credential:

Azure storage container:

URL prefix: https://<storageaccount>.blob.core.windows.net/

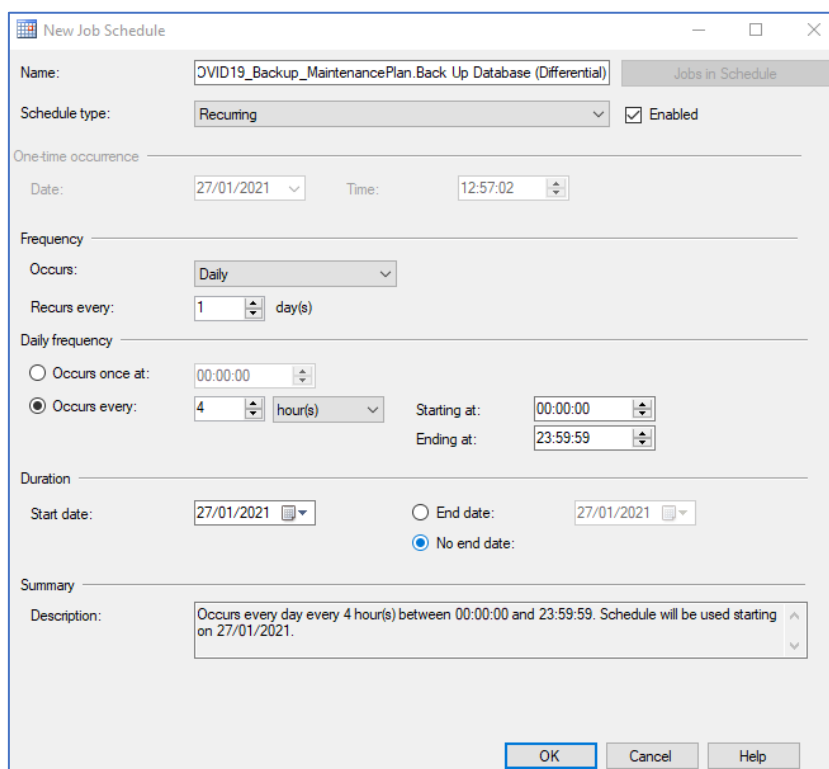
Backup file extension: bak

Schedule:
Occurs every day every 12 hour(s) between 00:00:00 and 23:59:59. Schedule v

Help < Back Next > Finish >> Cancel

Figura 65 - Passo 12

Cópias de Segurança (7)

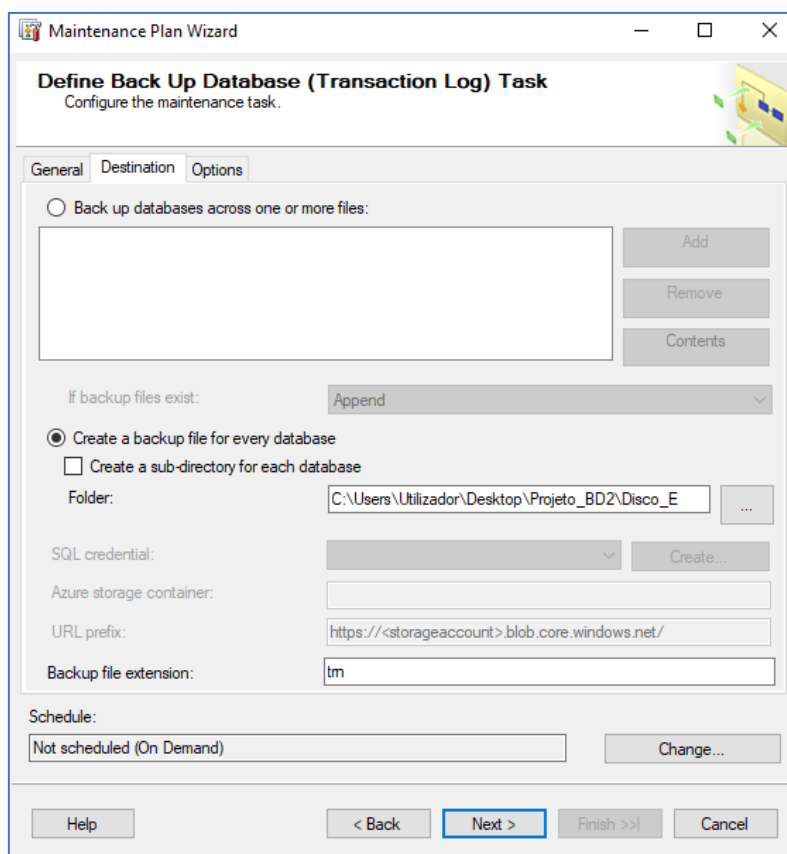


The 'New Job Schedule' dialog box is shown with the following configuration:

- Name:** COVID19_Backup_MaintenancePlan.Back Up Database (Differential)
- Schedule type:** Recurring
- Enabled:** ☒
- One-time occurrence:** Date: 27/01/2021, Time: 12:57:02
- Frequency:** Occurs: Daily, Recurs every: 1 day(s)
- Daily frequency:** ☒ Occurs every: 4 hour(s), Starting at: 00:00:00, Ending at: 23:59:59
- Duration:** Start date: 27/01/2021, ☐ End date: 27/01/2021, ☒ No end date
- Summary:** Description: Occurs every day every 4 hour(s) between 00:00:00 and 23:59:59. Schedule will be used starting on 27/01/2021.

Buttons: OK, Cancel, Help

Figura 66 - Passo 13



The 'Maintenance Plan Wizard' is shown at the 'Define Back Up Database (Transaction Log) Task' step. The 'General' tab is active.

Define Back Up Database (Transaction Log) Task
Configure the maintenance task.

General | Destination | Options

- ☐ Back up databases across one or more files:
- ☒ Create a backup file for every database
 - ☐ Create a sub-directory for each database
 - Folder: C:\Users\Utilizador\Desktop\Projeto_BD2\Disco_E
 - SQL credential: [dropdown] Create...
 - Azure storage container: [text box]
 - URL prefix: https://<storageaccount>.blob.core.windows.net/
 - Backup file extension: tm

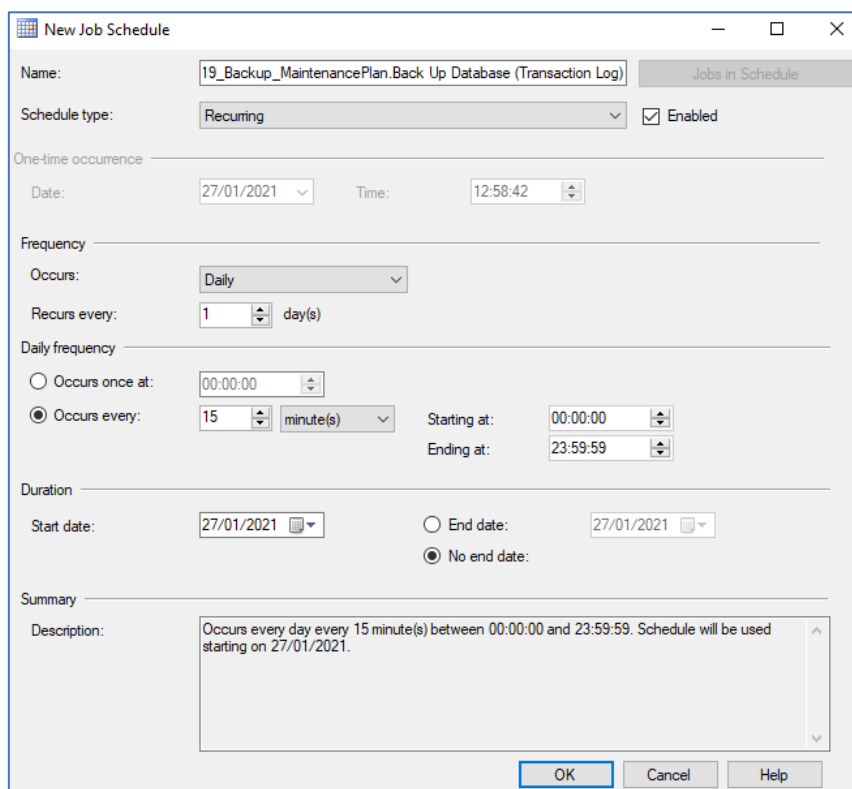
If backup files exist: Append

Schedule: Not scheduled (On Demand) Change...

Buttons: Help, < Back, Next >, Finish >>, Cancel

Figura 67 - Passo 14

Cópias de Segurança (8)

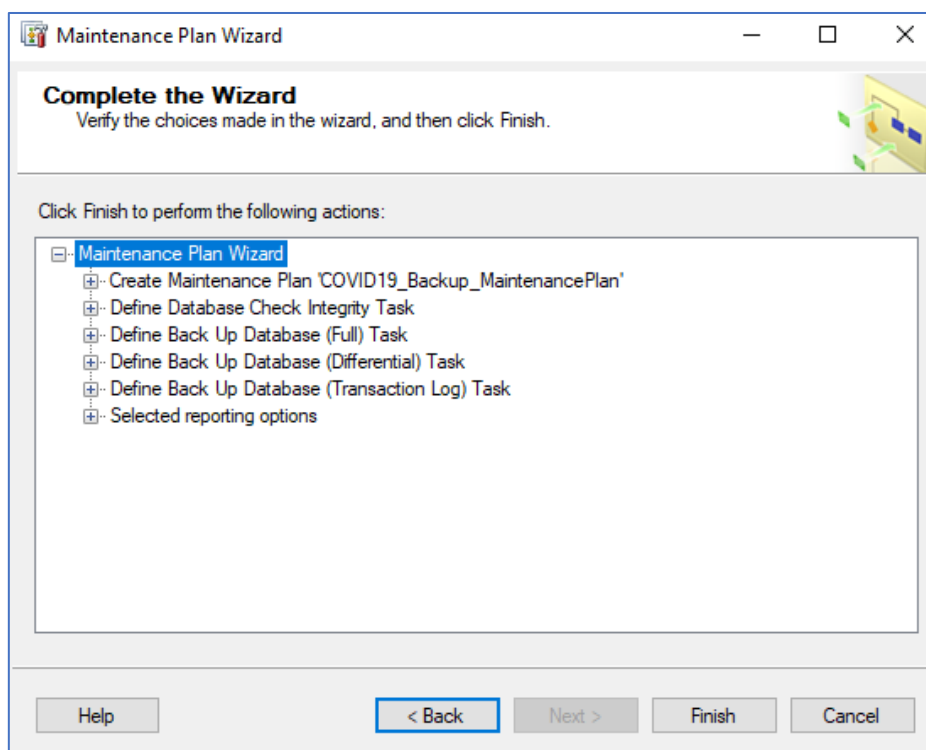


The 'New Job Schedule' dialog box is shown with the following configuration:

- Name:** 19_Backup_MaintenancePlan.Back Up Database (Transaction Log)
- Schedule type:** Recurring
- Enabled:** ☒
- One-time occurrence:** Date: 27/01/2021, Time: 12:58:42
- Frequency:** Occurs: Daily, Recurs every: 1 day(s)
- Daily frequency:** ☒ Occurs every: 15 minute(s), Starting at: 00:00:00, Ending at: 23:59:59
- Duration:** Start date: 27/01/2021, ☐ End date: 27/01/2021, ☒ No end date
- Summary:** Description: Occurs every day every 15 minute(s) between 00:00:00 and 23:59:59. Schedule will be used starting on 27/01/2021.

Buttons: OK, Cancel, Help

Figura 68 - Passo 15



The 'Maintenance Plan Wizard' window shows the 'Complete the Wizard' step. It instructs the user to 'Verify the choices made in the wizard, and then click Finish.' Below this, it lists the actions that will be performed by clicking Finish:

- Click Finish to perform the following actions:
- Maintenance Plan Wizard
 - Create Maintenance Plan 'COVID19_Backup_MaintenancePlan'
 - Define Database Check Integrity Task
 - Define Back Up Database (Full) Task
 - Define Back Up Database (Differential) Task
 - Define Back Up Database (Transaction Log) Task
 - Selected reporting options

Buttons: Help, < Back, Next >, Finish, Cancel

Figura 69 - Passo 16

Cópias de Segurança (9)

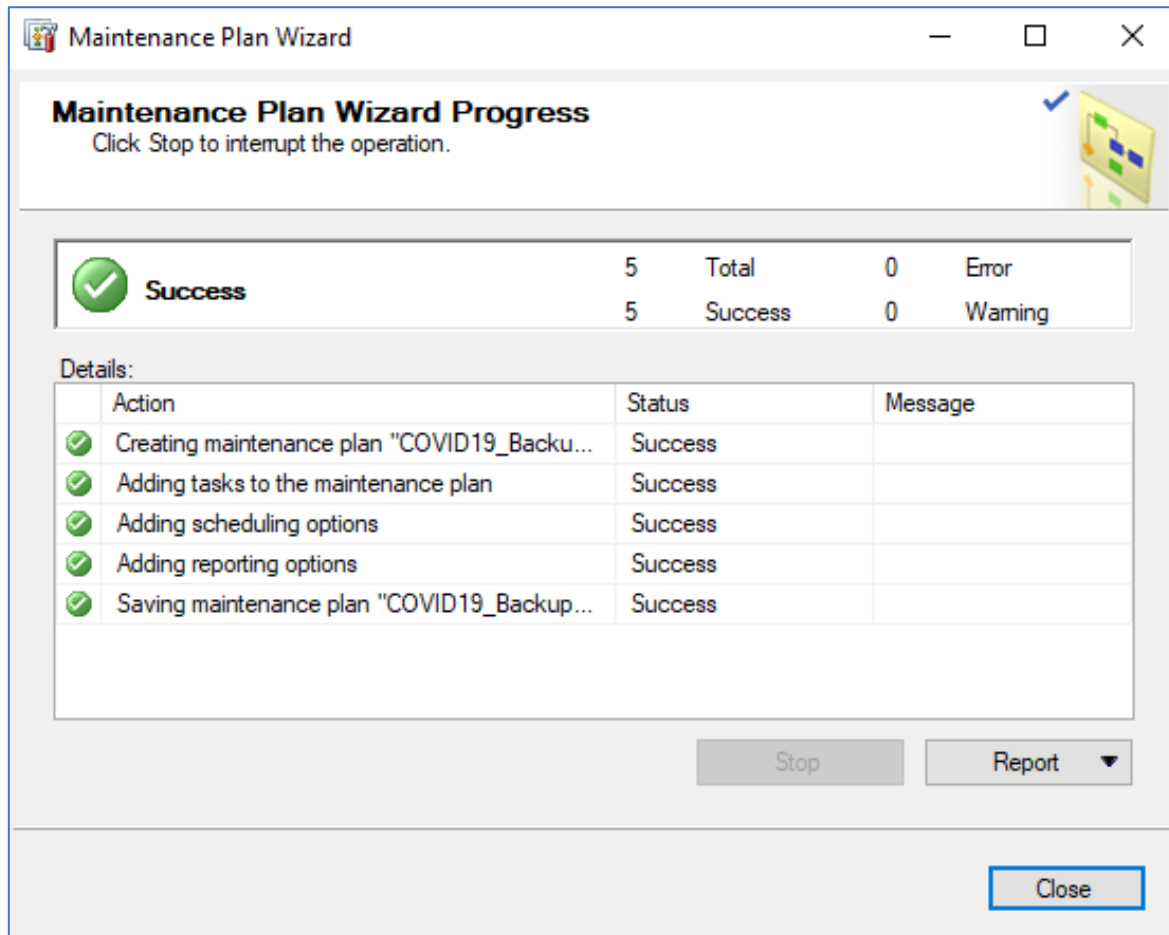


Figura 70 - Passo 17

Conclusão

Com a realização deste trabalho podemos concluir que as etapas da criação e gestão duma base de dados, independentemente da sua extensão, requerem devoção e a existência de um pensamento precedente antes da sua realização, uma vez que existe um elevado número de fatores implicados em todo o processo.

Para se proceder a uma automatização de uma base de dados tornando-a mais eficiente é possível atuar de diversas maneiras: Stored Procedures, Triggers ou mesmo a própria organização dos dados.

Outro aspeto que foi fulcral na realização do nosso trabalho foram as medidas de segurança tomadas e a proteção dos dados, que foi feita através de Backups.

Por fim, a realização deste trabalho foi benéfica, trazendo-nos múltiplos ganhos: foram exploradas as diferentes soluções que podem ser empregues nesta situação e experienciamos algumas das adversidades que podem surgir no nosso futuro trabalho, estando assim, mais preparados para lidar com as mesmas.

Bibliografia

Página da disciplina no moodle para esclarecimento de dúvidas:

<https://cms.ipbeja.pt/course/view.php?id=223>

Parte do código SQL para inserção de dados em algumas tabelas:

<https://github.com/KevMorelli/Country-Region-City-MSSQL>

Geração de Script de dados para tabela “Pessoas”:

<https://www.mockaroo.com>

Alguns dos sites consultados para esclarecimento de dúvidas ou correção de erros:

<https://stackoverflow.com/>

<https://www.w3schools.com/sql/>

<https://docs.microsoft.com/en-us/sql/>