



CrashJets

Empresa de Voos Charter

Microsoft SQL Server

Formador: João Paulo Santos

GRUPO

Cláudia Passarinho

Celso Sousa

José Carlos Silva

Miguel Sampayo Abrantes

14/06/2019

Índice

Levantamento de Requisitos e Níveis de Serviço	6
1. CrashJets – Empresa de Voos Charter	7
1.1. Levantamento dos requisitos	7
1.2. Descrição do problema	7
1.2.1. Introdução	7
1.2.2. Objetivos e níveis de serviço	7
1.3. Manual de Operação	8
1.3.1. Identificação dos objetos criados	8
1.3.2. Criação da Base de Dados	9
1.3.3. Diagrama Entidade-Associação (ERD)	10
1.3.4. Tabelas CrashJets	12
1.3.5. <i>User Defined Functions</i>	18
1.3.5.1. Udf_GetPrecoFinal	18
1.3.5.2. Udf_IsValidResData	18
1.3.5.3. Udf_IsVooDurMaior2H	19
1.3.5.4. Udf_AviaoEstaDisponivel	19
1.3.6. <i>User Stored Procedures</i>	19
1.3.6.1. Usp_ListaPassageiros	19
1.3.6.2. Usp_EfetuarReserva	20
1.3.6.3. Usp_Lotacao	20
1.4. Política de <i>Backups</i> e <i>Disaster Recovery</i>	21
1.4.1. Política de <i>Backups</i>	21
1.4.2. Política de <i>Disaster Recovery</i>	22
1.4.3. Plano de manutenção	26
1.4.4. Gestão de <i>Backups</i> e <i>Restores</i>	32
1.5. <i>Performance Tuning</i>	32
1.5.1. <i>Workload e Throughput</i>	40
1.5.2. Otimização	41
1.5.3. Monitorização	41
1.5.4. Recursos de Sistema	41
1.6. Permissões	42
1.6.1. <i>Users</i>	42
1.6.2. <i>Roles</i>	44
2. Código T-SQL	45
2.1. Criação da BD	45
2.2. Criação das Tabelas	46
2.2.1. Aviao	46
2.2.2. Refeicao	46
2.2.3. Passageiro	47

2.2.4.	Desconto.....	47
2.2.5.	Voo.....	48
2.2.6.	Escala	49
2.2.7.	Funcionario	49
2.2.8.	Reserva.....	50
2.3.	Adição das restrições.....	50
2.3.1.	Tabela aviao	50
2.3.2.	Tabela refeicao	51
2.3.3.	Tabela passageiro	51
2.3.4.	Tabela desconto.....	51
2.3.5.	Tabela voo.....	51
2.3.6.	Tabela escala	52
2.3.7.	Tabela funcionario.....	52
2.3.8.	Tabela reserva	53
2.4.	Funções desenvolvidas para a solução	54
2.4.1.	Cálculo do preço final da reserva	54
2.4.2.	Verificação se a data da reserva é anterior à data do voo	55
2.4.3.	Validação se a duração do voo é superior a 2 horas	55
2.4.4.	Validação da disponibilidade de um avião para ser utilizado num voo.....	56
2.5.	Procedimentos desenvolvidos para a solução.....	57
2.5.1.	Listagem completa de passageiros por voo	57
2.5.2.	Reserva de um voo com a validação de lotação do avião.....	58
2.5.3.	Obter horário e lotação de determinado voo	59
3.	Anexos.....	61
3.1.	SOLUÇÃO CRASHJETS – T-SQL.....	61
3.2.	Inserção dos dados nas tabelas – T-SQL	72
3.2.1.	Tabela aviao	72
3.2.2.	Tabela refeicao	73
3.2.3.	Tabela passageiro	73
3.2.4.	Tabela desconto.....	78
3.2.5.	Tabela voo.....	79
3.2.6.	Tabela escala	80
3.2.7.	Tabela funcionario.....	81
3.2.8.	Tabela reserva	84
3.3.	Ficheiros entregues.....	89

Índice de tabelas

Tabela 1 – Controlo de Versões	4
Tabela 2 – Acrónimos	4

Índice de figuras

Figura 1 – Espaço utilizado em disco.....	10
Figura 2 – Diagrama Entidade-Associação CrashJets (ERD)	11
Figura 3 – Tabelas, relações, campos e tipos de dados da solução CrashJets	12
Figura 4 – Criação de <i>job</i> para automatizar <i>backup</i>	23
Figura 5 – Criação de <i>job</i> para automatizar <i>backup</i> (continuação)	23
Figura 6 – Criação de <i>job</i> para automatizar <i>backup</i> (continuação)	24
Figura 7 – Criação de <i>job</i> para automatizar <i>backup</i> (continuação)	24
Figura 8 – Execução de <i>job</i> para <i>backup</i>	25
Figura 9 – <i>Job Log File Viewer</i>	25
Figura 10 – <i>Job Activity Monitor</i>	26
Figura 11 – Criação do Plano de Manutenção	27
Figura 12 – Criação do Plano de Manutenção (continuação).....	27
Figura 13 – Criação do Plano de Manutenção (continuação).....	28
Figura 14 – Execução do Plano de Manutenção	28
Figura 15 – Ficheiro de <i>backup</i> criado.....	28
Figura 16 – Relatório de <i>backup</i> gerado	29
Figura 17 – Relatório de <i>backup</i> gerado (conteúdo)	29
Figura 18 – Criação de alerta para controlo do tamanho do <i>log file</i>	30
Figura 19 – Criação de alerta para controlo do tamanho do <i>log file</i> (continuação).....	30
Figura 20 – Criação de alerta para controlo do tamanho do <i>log file</i> (continuação).....	31
Figura 21 – Criação de alerta para controlo do tamanho do <i>data file</i>	32
Figura 22 – Monitor de desempenho (<i>counters</i>)	33
Figura 23 – Monitor de desempenho (gráfico após execução da <i>stored procedure usp_ListaPassageiros</i>).....	34
Figura 24 – Criação de <i>Stress Test</i>	34
Figura 25 – Criação de <i>Stress Test</i> (continuação)	35
Figura 26 – Criação de <i>Stress Test</i> (<i>counters</i>).....	35
Figura 27 – Criação de <i>Stress Test</i> (continuação)	36
Figura 28 – Criação de <i>Stress Test</i> (continuação)	36
Figura 29 – Execução do <i>Stress Test</i> criado.....	37
Figura 30 – Execução do <i>Stress Test</i> criado.....	37
Figura 31 – Execução de <i>stress query</i> à base de dados CrashJets	38
Figura 32 – Execução do <i>Stress Test</i> criado.....	38
Figura 33 – Localização do <i>log file</i> resultado da execução do <i>Stress Test</i> criado.....	38
Figura 34 – <i>Log file</i> após execução da <i>stress query</i> à base de dados CrashJets	39
Figura 35 – <i>Trace CrashJetsTrace.trc</i> (localização).....	39
Figura 36 – Propriedades do <i>trace CrashJetsTrace</i>	40
Figura 37 – Confirmação dos resultados do <i>trace CrashJetsTrace</i>	40
Figura 38 – Criação de conta de utilizador Windows de administração	42

Figura 39 – Criação de conta de utilizador Windows de negócio	42
Figura 40 – Contas de utilizador Windows criadas.....	43
Figura 41 – Criação de novos logins no SQL Server para os utilizadores Windows criados	43
Figura 42 – Atribuição do <i>role</i> de servidor <i>sysadmin</i> ao <i>login</i> do utilizador de administração	44
Figura 43 – Atribuição do <i>role</i> da base de dados CrashJets <i>db_executor</i> ao <i>login</i> de um dos utilizadores de negócio.....	45

Controlo de versões

Versão	Estado	Data	Sumário das alterações	Autores
0.01	Versão inicial	14/06/2019	Versão inicial	Cláudia Passarinho Celso Sousa José Carlos Silva Miguel Sampayo Abrantes
1.00	Versão revista	xx/06/2019	-	■

Tabela 1 – Controlo de Versões

Acrónimos

	Definição
3FT	Terceira Formula Normal
BAK	Backup
BD	Base de Dados
ERD	Entity Relationship Diagram
LDF	Log Database File
LOG	Log de dados
MDF	Master Database File
N/A	Não se aplica
SQL	Structured Query Language
T-SQL	Transact-SQL
Trigger	Procedimento armazenado em DB para ser invocado de forma automática
UDDT	User-defined data type
UDT	User-defined type
UDF	User Defined Functions

Tabela 2 – Acrónimos

Levantamento de Requisitos e Níveis de Serviço

Objetivo

O objetivo deste documento visa enunciar as atividades e configurações realizadas no decorrer do trabalho de grupo relacionadas com o projeto de implementação em SQL Server do exercício prático CrashJets, Empresa de Voos Charter, no âmbito do curso de programação em Microsoft SQL Server do Citeforma.

Todo o documento é suportado pela aplicação do conhecimento adquirido nas aulas e experiência profissional que visa a utilização das melhores práticas, alimentadas pela experiência que conduzem à melhoria dos níveis de serviço com base nas tecnologias em uso.

Âmbito

Este documento apresenta a solução para a empresa que tem como target de mercado da aviação, voos económicos a operar na região da Europa Ocidental.

- A sua frota é constituída, atualmente por 20 aviões, com paragem de 1 dia, por mês, em oficina para manutenção programada. A frota assenta em 3 tipos de modelo com capacidade de 80, 120 e 140 passageiros.
- Como requisito não é permitindo a venda de bilhetes acima da capacidade de lugares.
- A companhia tem 120 funcionários que se dividem em categorias profissionais: tripulação, assistentes de embarque e administrativos.
- Por cada voo, os clientes da companhia têm direito a uma refeição ligeira nos voos de duração superior a 2 horas, escolhida no ato da reserva. As opções são: Dieta, Normal e Vegetariana.
- A companhia dispõe de um programa de pontos que beneficia os seus clientes habituais, através de milhas e descontos em viagens.

Audiência

Este documento destina-se ao docente da Unidade Curricular com vista de avaliar e aprovar o conteúdo aqui descrito.

1. CrashJets – Empresa de Voos Charter

1.1. Levantamento dos requisitos

Solução a ser desenvolvida em SQL Server:

- **Frota:** constituída por 20 aviões, com paragem de um dia por mês para manutenção programada
- **Capacidade da frota:** 3 tipos de modelos, com capacidade de 80, 120 e 140 passageiros
- **Venda de bilhetes:** Não é permitido a emissão de bilhetes acima da capacidade máxima de lugares
- **Colaboradores:** A companhia tem 120 funcionários, subdivididos em 3 categorias profissionais: Tripulação, Assistentes de Embarque e Administradores
- **Refeições:** Os clientes podem escolher uma refeição ligeira nos voos de duração superior a 2 horas, escolhida no ato da reserva. Tipos de refeições: Dieta, Normal e Vegetariana
- **Política de pontos e descontos:** A companhia dispõe de pontos que beneficia os seus clientes habituais através de milhas e descontos em viagens
- **Cliente/Servidor:** Utilizar as características e funcionalidades da linguagem SQL, tomando em consideração a normalização dos dados, segurança e permissões do acesso de utilizadores
- **Programação:** T-SQL, Stored Procedures e Functions

1.2. Descrição do problema

1.2.1. Introdução

Este documento apresenta a solução para a empresa que tem como *target* de mercado da aviação, voos económicos a operar na região da Europa Ocidental.

A sua frota é constituída, atualmente por 20 aviões, com paragem de 1 dia, por mês, em oficina para manutenção programada. A frota assenta em 3 tipos de modelo com capacidade de 80, 120 e 140 passageiros.

Como requisito não é permitindo a venda de bilhetes acima da capacidade de lugares.

A companhia tem 120 funcionários que se dividem em categorias profissionais: tripulação, assistentes de embarque e administrativos.

Por cada voo, os clientes da companhia têm direito a uma refeição ligeira nos voos de duração superior a 2 horas, escolhida no ato da reserva. As opções são: Dieta, Normal e Vegetariana.

A companhia dispõe de um programa de pontos que beneficia os seus clientes habituais, através de milhas e descontos em viagens.

1.2.2. Objetivos e níveis de serviço

Pretende-se implementar uma base de dados em SQL Server, com os mecanismos de segurança, operação e de consulta de dados.

É solicitado um modelo ERD (*Entity Relationship Diagram*) normalizado, cumprindo as regras do negócio da companhia. Esta base de dados deverá dispor de três *filegroups* (*primary datafile*, *secondary datafile* e *log file*).

Criação de *users* e *roles* necessários ao seu normal funcionamento com as permissões e mecanismos de auditoria.

Implementação de uma política de *backup* e *restore* que inclua *devices* para *backup*, plano de *backup* diário e plano para *restore*.

Implementação de um mecanismo de monitorização e *performance tuning*.

A solução inclui as seguintes operações:

- 1. Listagem Completa de Passageiros por voo.
- 2. Transação de reserva de um voo com validação de lotação do avião.
- 3. Consulta de horário e lotação de determinado voo.

Pretende-se que seja garantido a disponibilidade, a eficiência e a fiabilidade no acesso e utilização da base de dados.

1.3. Manual de Operação

1.3.1. Identificação dos objetos criados

A estrutura da base de dados criada denomina-se por *CrashJets*. É composta pelos seguintes ficheiros de *data space* e *log space*:

- O *primary filegroup* com um *primary data file* com *filename* **CrashJets_Principal.mdf**, com *logical_file_name* **CrashJets_Primary**, na diretoria **C:\CrashJets**, com tamanho inicial de **128MB**, tamanho máximo de **256MB** e *autogrowth* **desativado**;
- O *secondary filegroup* com um *secondary data file* com *filename* **MyCrashJets_Secondary01.ndf**, com *logical_file_name* **CrashJets_Secondary01**, na diretoria **C:\CrashJets**, com tamanho inicial de **128MB**, tamanho máximo de **256MB** e *autogrowth* **desativado**;
- O *log file* com *filename* **CrashJets.ldf**, com *logical_file_name* **CrashJets_Log** na diretoria **C:\CrashJets**, com tamanho inicial de **1024MB**, tamanho máximo de **2048MB** e *autogrowth* **desativado**.

Utilizando as *stored procedures* abaixo indicadas é possível mostrar as informações relativas a estes *filegroups*:

EXEC sp_helpfile;

	name	fileid	filename	filegroup	size	maxsize	growth	usage
1	CrashJets_Primary	1	C:\CrashJets\CrashJets_Principal.mdf	PRIMARY	131072 KB	262144 KB	0 KB	data only
2	CrashJets_Log	2	C:\CrashJets\CrashJets.ldf	NULL	1048576 KB	2097152 KB	0 KB	log only
3	C:\CrashJets\CrashJets_Secondary01	3	C:\CrashJets\MyCrashJets_Secondary01.ndf	CrashJets_Filegroup	131072 KB	262144 KB	0 KB	data only

EXEC sp_helpfile 'CrashJets_Primary';

	name	filename	filegroup	size	maxsize	growth	usage
1	CrashJets_Primary	C:\CrashJets\CrashJets_Principal.mdf	PRIMARY	131072 KB	262144 KB	0 KB	data only

EXEC sp_helpfile 'C:\CrashJets\CrashJets_Secondary01';

	name	filename	filegroup	size	maxsize	growth	usage
1	C:\CrashJets\CrashJets_Secondary01	C:\CrashJets\MyCrashJets_Secondary01.ndf	CrashJets_Filegroup	131072 KB	262144 KB	0 KB	data only

EXEC sp_helpfile 'CrashJets_Log';

	name	filename	filegroup	size	maxsize	growth	usage
1	CrashJets_Log	C:\CrashJets\CrashJets.ldf	NULL	1048576 KB	2097152 KB	0 KB	log only

EXEC sp_helpfilegroup;

	groupname	groupid	filecount
1	PRIMARY	1	1
2	CrashJets_Filegroup	2	1

EXEC sp_helpfilegroup 'Primary';

	groupname	groupid	filecount
1	PRIMARY	1	1

	file_in_group	fileid	filename	size	maxsize	growth
1	CrashJets_Primary	1	C:\CrashJets\CrashJets_Principal.mdf	131072 KB	262144 KB	0 KB

```
EXEC sp_helpfilegroup 'CrashJets_Filegroup';
```

	groupname	groupid	filecount
1	CrashJets_Filegroup	2	1

	file_in_group	fileid	filename	size	maxsize	growth
1	C:\CrashJets\CrashJets_Secondary01	3	C:\CrashJets\MyCrashJets_Secondary01.ndf	131072 KB	262144 KB	0 KB

Optámos pelo aumento de tamanho máximo dos *filegroups* em modo manual e por um tamanho inicial dos *filegroups* com maior margem de desenvolvimento.

O ficheiro de *log* tem tamanho inicial de **1024MB** e tamanho máximo quatro vezes superior à soma dos tamanhos máximos definidos para os *primary data file* e *secondary data file* e duas vezes superior ao tamanho inicial do *log file*.

Desta forma garante-se que não haja a necessidade de frequentes incrementos no tamanho atual do *log file*, melhorando a eficiência da base de dados e minimizando:

- A interferência na velocidade de execução das *queries* durante o processo de aumento do tamanho atual do *log file*;
- *Backups* e *restores* mais lentos;
- Arranque mais lento do serviço do SQL Server.

O tamanho máximo do *log file* foi escolhido de forma a prevenir que a base de dados fique inacessível no caso de o ficheiro chegar ao seu limite máximo de tamanho. Pretende-se proteger a base de dados e garantir a qualidade e eficiência do trabalho do *Database Administrator* que deve monitorizar frequentemente o nível de utilização do *log file* e o espaço livre disponível.

No ponto seguinte são apresentadas as tabelas e respetivos campos da base de dados CrashJets.

1.3.2. Criação da Base de Dados

É necessário que o sistema utilizado para armazenar a solução inclua instalado o MS SQL Server 2017 Developer Edition e o SQL Server Management Studio 2017 para se proceder à criação da base de dados e utilização da solução CrashJets.

Nestas condições de ambiente de trabalho, a BD pode ser criada através da utilização da instrução *Create Database* em T-SQL, em anexo. Serão criadas automaticamente o *data file* e *log file*. O utilizador que cria a BD deverá ter todas as permissões para usar a *Master DB*, para se poder registar nas tabelas **sysdatabases** e **sysaltfiles** a informação das novas tabelas da solução.

Na criação da BD é definida a seguinte informação:

- **Primary File:** CrashJets_Primary
 - **File Name and Location:** C:\CrashJets\CrashJets_Principal.mdf
 - **Tamanho inicial:** 128MB
 - **Tamanho Máximo:** 256MB
 - **File Growth:** 0
- **Secondary File:** CrashJets_Secondary01
 - **File Name and Location:** C:\CrashJets\MyCrashJets_Secondary01.ndf

- **Tamanho inicial:** 128MB
- **Tamanho Máximo:** 256MB
- **File Growth:** 0
- **Transaction Log:** CrashJets_Log
 - **LOG File Name and Location:** C:\CrashJets\CrashJets.ldf
 - **LOG Size:** 1024MB
 - **Maxsize:** 2048MB
 - **File Growth:** 0

Em seguida, apresenta-se informação sobre o espaço em disco ocupado pela base de dados:

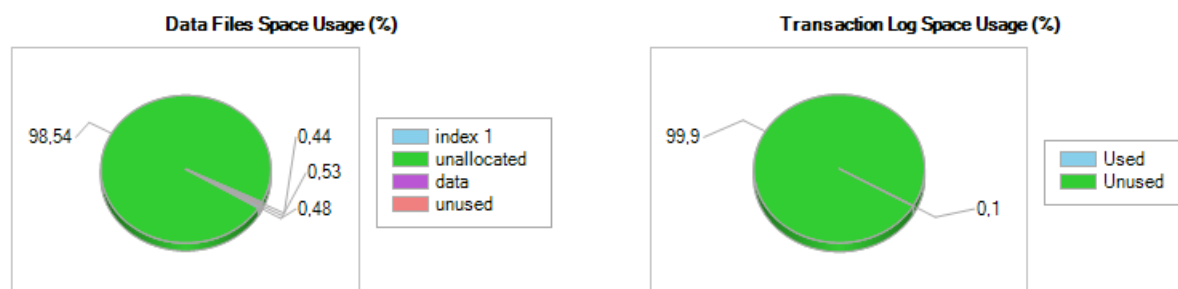
`exec sp_spaceused;`

	database_name	database_size	unallocated space
1	CrashJets	1280.00 MB	252.09 MB

	reserved	data	index_size	unused
1	4008 KB	1200 KB	1264 KB	1544 KB

This report provides overview of the utilization of disk space within the Database.

Total Space Reserved	1,25 GB
Data Files Space Reserved	256,00 MB
Transaction Log Space Reserved	1 024,00 MB



No entry found for autogrow/autoshrink event for CrashJets database in the trace log.

□ Disk Space Used by Data Files

Filegroup Name	Logical File Name	Physical File Name	Space Reserved	Space Used
CrashJets_Filegroup	C:\CrashJets\CrashJets_Secondary01	C:\CrashJets\MyCrashJets_Secondary01.ndf	128,00 MB	320,00 KB
PRIMARY	CrashJets_Primary	C:\CrashJets\CrashJets_Principal.mdf	128,00 MB	3,81 MB

Figura 1 – Espaço utilizado em disco

1.3.3. Diagrama Entidade-Associação (ERD)

A solução inclui a BD relacional que inclui tabelas que representam o sujeito/entidade por tabela, que são unidas entre si através das relações. Cada tabela contém campos/atributos e respetivos tipos de dados, uma *primary key* e eventualmente uma ou mais *foreign key* (chaves estrangeiras/externas). Inclui ainda *user defined functions* e *user stored procedures* necessárias à implementação dos requisitos do projeto.

De seguida, são apresentados os vários objetos constituintes.

Fez-se uso da 3FN para normalizar e otimizar a performance da BD. O desenho do processo consistiu em remover as redundâncias, para remover a repetição dos dados nas diferentes tabelas, com a exceção das chaves que são usadas para unir as entidades.

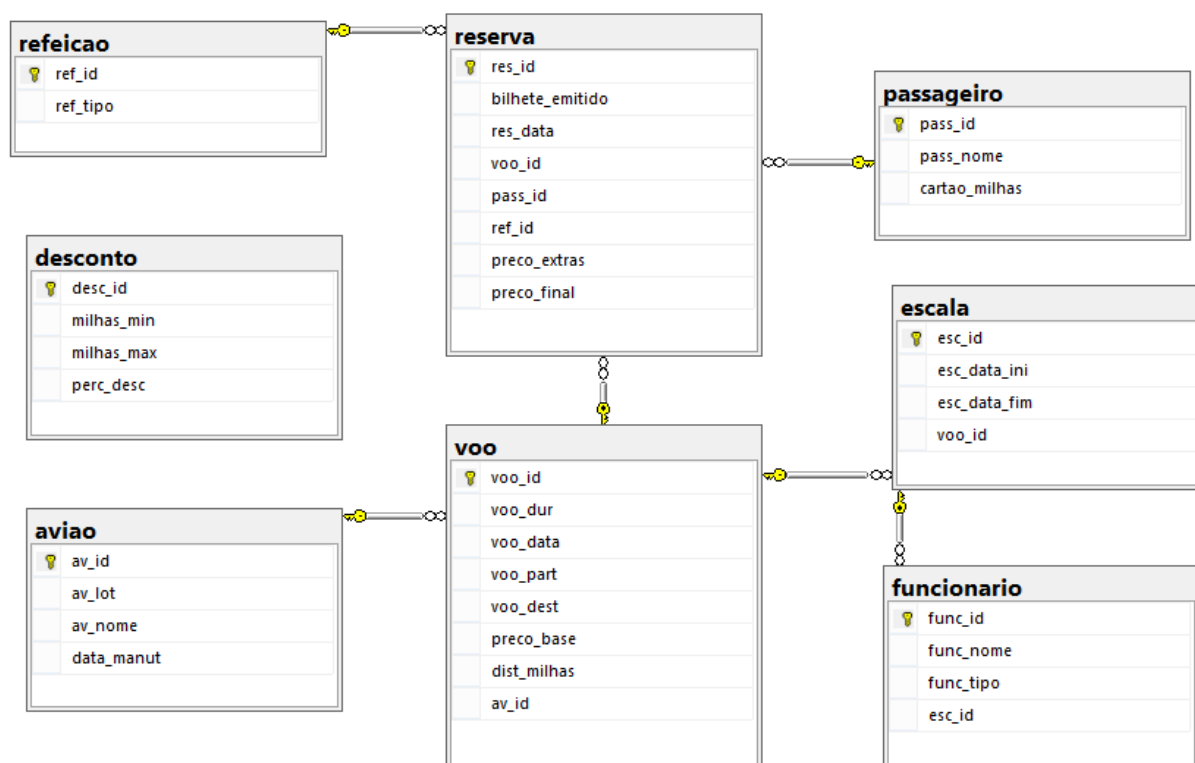


Figura 2 – Diagrama Entidade-Associação CrashJets (ERD)

A solução é constituída por 8 entidades, relacionadas entre si, nomeadamente:

- **Funcionário e Escala:** Um funcionário tem uma escala. Uma escala pode ser constituída por vários funcionários
- **Voo e Escala:** Uma escala corresponde a um voo. Um voo pode ter várias escalas
- **Voo e Reserva:** Um voo pode ter várias reservas. Uma reserva pertence a um só voo
- **Voo e Avião:** Um voo é realizado por um avião. Um avião pode realizar vários voos
- **Reserva e Refeição:** Uma reserva pode ter direito a uma refeição ou não ter direito a nenhuma refeição no caso de voos de duração não superior a duas horas. A mesma refeição pode ser escolhida por várias reservas distintas

1.3.4. Tabelas CrashJets



Figura 3 – Tabelas, relações, campos e tipos de dados da solução CrashJets

Através dos seguintes comandos demonstramos a constituição das oito tabelas criadas, respetivos campos e *constraints*:

EXEC sp_help 'aviao';

1	aviao	dbo	user table	2019-06-10 10:53:04.287						
	Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource	Collation
1	av_id	int	no	4	10	0	no	(n/a)	(n/a)	NULL
2	av_lot	int	no	4	10	0	no	(n/a)	(n/a)	NULL
3	av_nome	nvarchar	no	500			no	(n/a)	(n/a)	Latin1_General_CI_AS
4	data_manut	date	no	3	10	0	no	(n/a)	(n/a)	NULL
	Identity	Seed	Increment	Not For Replication						
1	av_id	1	1	0						
	RowGuidCol									
1	No rowguidcol column defined.									
	Data located on filegroup									
1	PRIMARY									
	index_name	index_description					index_keys			
1	aviao_av_id_pk	clustered, unique, primary key located on PRIMARY					av_id			
	constraint_type	constraint_name		delete_action	update_action	status_enabled	status_for_replication	constraint_keys		
1	PRIMARY KEY (clustered)	aviao_av_id_pk		(n/a)	(n/a)	(n/a)	(n/a)	av_id		
2	CHECK on column av_lot	aviao_av_lot_ck		(n/a)	(n/a)	Enabled	Is_For_Replication	([av_lot]=(140) OR [av_lot]=(120) OR [av_lot]=(80))		
3	CHECK on column data_manut	aviao_data_manut_ck		(n/a)	(n/a)	Enabled	Is_For_Replication	([data_manut]>CONVERT([date],getdate()))		
4	CHECK on column av_id	aviao_maximo_avioes_ck		(n/a)	(n/a)	Enabled	Is_For_Replication	([av_id]<=(20))		
	Table is referenced by foreign key									
1	CrashJets.dbo.voo: voo_av_id_fk									

- Tabela: *aviao*;
- Campos:
 - *av_id*: identificador; tipo **int**
 - *av_lot*: lotação; tipo **int**
 - *av_nome*: nome descritivo do avião; tipo **nvarchar(250)**
 - *data_manut*: data de manutenção programada; tipo **date**
- *Constraints*:
 - Os quatro campos desta tabela não aceitam o valor *null*;
 - O campo *av_id* é a chave primária;
 - O campo *av_lot* tem de ter um de três valores de lotação: 80, 120 ou 140;
 - O campo *av_id* está limitado ao valor de 20 que é o número máximo da frota de aviões;

- O campo *data_manut* tem de ter uma data superior à data do dia atual.

EXEC sp_help 'refeicao';

Name	Owner	Type	Created_datetime	
1	refeicao	dbo	user table	2019-06-10 10:53:04.293

Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource	Collation	
1	ref_id	int	no	4	10	0	no	(n/a)	(n/a)	NULL
2	ref_tipo	nvarchar	no	200			no	(n/a)	(n/a)	Latin1_General_CI_AS

Identity	Seed	Increment	Not For Replication	
1	ref_id	1	1	0

RowGuidCol	
1	No rowguidcol column defined.

Data_located_on_filegroup	
1	PRIMARY

index_name	index_description	index_keys	
1	refeicao_ref_id_pk	clustered, unique, primary key located on PRIMARY	ref_id

constraint_type	constraint_name	delete_action	update_action	status_enabled	status_for_replication	constraint_keys	
1	PRIMARY KEY (clustered)	refeicao_ref_id_pk	(n/a)	(n/a)	(n/a)	ref_id	
2	CHECK on column ref_tipo	refeicao_ref_tipo_ck	(n/a)	(n/a)	Enabled	Is_For_Replication	((ref_tipo)='Vegetariana' OR (ref_tipo)='Dieta' OR (ref_tipo)='Normal')

Table is referenced by foreign key	
1	CrashJets.dbo.reserva: reserva_ref_id_fk

- Tabela: *refeicao*;
- Campos:
 - *ref_id*: identificador, tipo **int**
 - *ref_tipo*: tipo de refeição, tipo **nvarchar(100)**
- Constraints:
 - Os dois campos desta tabela não aceitam o valor *null*;
 - O campo *ref_id* é a chave primária;
 - O campo *ref_tipo* tem de ter um de três valores de tipo de refeição: 'Normal', 'Vegetariana' ou 'Dieta'.

EXEC sp_help 'passageiro';

	Name	Owner	Type	Created_datetime
1	passageiro	dbo	user table	2019-06-10 10:53:04.303

	Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource	Collation
1	pass_id	int	no	4	10	0	no	(n/a)	(n/a)	NULL
2	pass_nome	nvarchar	no	200			no	(n/a)	(n/a)	Latin1_General_CI_AS
3	cartao_milhas	int	no	4	10	0	no	(n/a)	(n/a)	NULL

	Identity	Seed	Increment	Not For Replication
1	pass_id	1	1	0

	RowGuidCol
1	No rowguidcol column defined.

	Data_located_on_filegroup
1	PRIMARY

	index_name	index_description	index_keys
1	passageiro_pass_id_pk	clustered, unique, primary key located on PRIMARY	pass_id

	constraint_type	constraint_name	delete_action	update_action	status_enabled	status_for_replication	constraint_keys
1	CHECK on column cartao_milhas	desconto_cartao_milhas_ck	(n/a)	(n/a)	Enabled	Is_For_Replication	((cartao_milhas)>=(0))
2	PRIMARY KEY (clustered)	passageiro_pass_id_pk	(n/a)	(n/a)	(n/a)	(n/a)	pass_id

	Table is referenced by foreign key
1	CrashJets.dbo.reserva: reserva_pass_id_fk

- Tabela: *passageiro*;
- Campos:
 - *pass_id*: identificador, tipo **int**
 - *pass_nome*: nome do passageiro, tipo **nvarchar(100)**

- *cartao_milhas*: valor relativo ao somatório das milhas dos voos reservados (com pagamento efetuado) pelo passageiro até ao momento presente. De notar que, no caso de uma reserva para um determinado voo ser desmarcada (com devolução do valor pago pelo cliente), teria de ser criada uma *stored procedure* que subtraísse o valor de milhas relativo ao voo dessa reserva entretanto cancelada, tipo **int**
- *Constraints*:
 - Os três campos desta tabela não aceitam o valor *null*;
 - O campo *pass_id* é a chave primária;
 - O campo *cartao_milhas* tem de ter valor um valor não negativo.

EXEC sp_help 'desconto';

	Name	Owner	Type	Created_datetime
1	desconto	dbo	user table	2019-06-10 10:53:04.310

	Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource	Collation
1	desc_id	int	no	4	10	0	no	(n/a)	(n/a)	NULL
2	milhas_min	bigint	no	8	19	0	no	(n/a)	(n/a)	NULL
3	milhas_max	bigint	no	8	19	0	no	(n/a)	(n/a)	NULL
4	perc_desc	int	no	4	10	0	no	(n/a)	(n/a)	NULL

	Identity	Seed	Increment	Not For Replication
1	No identity column defined.	NULL	NULL	NULL

	RowGuidCol
1	No rowguidcol column defined.

	Data_located_on_filegroup
1	PRIMARY

	index_name	index_description	index_keys
1	desconto_desc_id_pk	clustered, unique, primary key located on PRIMARY	desc_id

	constraint_type	constraint_name	delete_action	update_action	status_enabled	status_for_replication	constraint_keys
1	PRIMARY KEY (clustered)	desconto_desc_id_pk	(n/a)	(n/a)	(n/a)	(n/a)	desc_id
2	CHECK Table Level	desconto_milhas_max_ck	(n/a)	(n/a)	Enabled	Is_For_Replication	((milhas_max)>(milhas_min))
3	CHECK on column milhas_min	desconto_milhas_min_ck	(n/a)	(n/a)	Enabled	Is_For_Replication	((milhas_min)>=(1000))

- Tabela: *desconto*;
- Campos:
 - *desc_id*: identificador, tipo **int**
 - *milhas_min*: limite mínimo de milhas para uma determinada percentagem de desconto, tipo **bigint**
 - *milhas_max*: limite máximo de milhas para uma determinada percentagem de desconto, tipo **bigint**
 - *perc_desc*: percentagem de desconto a aplicar no preço final da reserva determinada pelo intervalo de milhas em que se situa o valor do *cartao_milhas* do passageiro, tipo **int**
- *Constraints*:
 - Os quatro campos desta tabela não aceitam o valor *null*;
 - O campo *desc_id* é a chave primária;
 - O campo *milhas_max* tem de ter um valor superior ao valor do campo *milhas_min*;
 - O campo *milhas_min* tem de ter um valor maior ou igual a 1000 para que possa haver direito a uma percentagem de desconto no preço final da reserva.

EXEC sp_help 'voo';

SQLQuery2.sql - M... (MEUPC,Celso (56))

EXEC sp_help 'voo';

100 %

Name	Owner	Type	Created_date
voo	dbo	user table	2019-06-10 10:53:04.317

Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource	Collation
voo_id	int	no	4	10	0	no	(n/a)	(n/a)	NULL
voo_dur	time	no	5	16	7	no	(n/a)	(n/a)	NULL
voo_data	datetime	no	8			no	(n/a)	(n/a)	NULL
voo_part	nvarchar	no	200			no	(n/a)	(n/a)	Latin1_General_CI_AS
voo_dest	nvarchar	no	200			no	(n/a)	(n/a)	Latin1_General_CI_AS
preco_base	int	no	4	10	0	no	(n/a)	(n/a)	NULL
dist_milhas	int	no	4	10	0	no	(n/a)	(n/a)	NULL
av_id	int	no	4	10	0	no	(n/a)	(n/a)	NULL

Identity	Seed	Increment	Not For Replication
voo_id	1	1	0

RowGuidCol

1 No rowguidcol column defined.

Data located on filegroup

1 PRIMARY

index_name	index_description	index_keys
voo_voo_id_pk	clustered, unique, primary key located on PRIMARY	voo_id

constraint_type	constraint_name	delete_action	update_action	status_enabled	status_for_replication	constraint_keys
FOREIGN KEY	voo_av_id_fk	No Action	No Action	Enabled	Is_For_Replication	av_id
CHECK Table Level	voo_aviao_disponivel_ck	(n/a)	(n/a)	Enabled	Is_For_Replication	REFERENCES CrashJets.dbo.aviao (av_id)
CHECK on column dist_milhas	voo_dist_milhas_ck	(n/a)	(n/a)	Enabled	Is_For_Replication	[(dbo].[udf_AviaoEstaDisponivel])(av_id,voo_id)=1)
CHECK on column preco_base	voo_preco_base_ck	(n/a)	(n/a)	Enabled	Is_For_Replication	[(dist_milhas)>0]
CHECK on column voo_data	voo_voo_data_ck	(n/a)	(n/a)	Enabled	Is_For_Replication	[(preco_base)>0]
PRIMARY KEY (clustered)	voo_voo_id_pk	(n/a)	(n/a)	(n/a)	(n/a)	voo_id

Table is referenced by foreign key

1 CrashJets.dbo.escala: escala_voo_id_fk

2 CrashJets.dbo.reserva: reserva_voo_id_fk

Query executed successfully.

MEUPC (14.0 RTM) MEUPC/Celso (56) CrashJets 00:00:02 2 rows

- Tabela: *voo*;
- Campos:
 - *voo_id*: identificador, tipo **int**
 - *voo_dur*: duração do voo, tipo **time**
 - *voo_data*: data e hora do voo, tipo **datetime**
 - *voo_part*: local de partida do voo, tipo **nvarchar(100)**
 - *voo_dest*: local de destino do voo, tipo **nvarchar(100)**
 - *preco_base*: valor base do preço do voo ao qual pode ser somado um valor de extras não contemplados no preço base (por exemplo, de bagagem) e subtraída uma percentagem do preço final de acordo com o *cartao_milhas* do passageiro, tipo **int**
 - *dist_milhas*: distância do voo em milhas, tipo **int**
 - *av_id*: identificador da tabela *aviao* em cujo qual se realizará o voo, tipo **int**
- Constraints:
 - Os oito campos desta tabela não aceitam o valor *null*;
 - O campo *voo_id* é a chave primária;
 - O campo *av_id* é chave estrangeira que relaciona a tabela *voo* com a tabela *aviao*;
 - O campo *dist_milhas* tem de ter um valor de distância em milhas do voo superior a 0;
 - O campo *preco_base* tem de ter um valor de preço base do voo superior a 0;
 - O campo *voo_data* tem de ter uma data superior à data atual;
 - É verificada a disponibilidade do avião para o voo através da chamada de uma função denominada *udf_AviaoEstaDisponivel* que recebe como parâmetros o *av_id* e o *voo_id* e verifica se o avião com o *av_id* se encontra em manutenção ou não (caso se encontre em manutenção na data do voo *voo_id* não poderá ser o aparelho utilizado para o voo).

EXEC sp_help 'escala';

Name	Owner	Type	Created_datetime	
1	escala	dbo	user table	2019-06-10 10:53:04.323

Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource	Collation	
1	esc_id	int	no	4	10	0	no	(n/a)	(n/a)	NULL
2	esc_data_ini	datetime	no	8			no	(n/a)	(n/a)	NULL
3	esc_data_fim	datetime	no	8			no	(n/a)	(n/a)	NULL
4	voo_id	int	no	4	10	0	no	(n/a)	(n/a)	NULL

Identity	Seed	Increment	Not For Replication	
1	esc_id	1	1	0

RowGuidCol	
1	No rowguidcol column defined.

Data_located_on_filegroup	
1	PRIMARY

index_name	index_description	index_keys	
1	escala_esc_id_pk	clustered, unique, primary key located on PRIMARY	esc_id

constraint_type	constraint_name	delete_action	update_action	status_enabled	status_for_replication	constraint_keys	
1	CHECK Table Level						
2	PRIMARY KEY (clustered)	escala_esc_data_fim_ck	(n/a)	(n/a)	Enabled	Is_For_Replication	((esc_data_fim)>[esc_data_ini])
3	FOREIGN KEY	escala_esc_id_pk	(n/a)	(n/a)	(n/a)	(n/a)	esc_id
4		escala_voo_id_fk	No Action	No Action	Enabled	Is_For_Replication	voo_id
							REFERENCES CrashJets.dbo.voo (voo_id)

Table is referenced by foreign key	
1	CrashJets.dbo.funcionario: funcionario_esc_id_fk

- Tabela: *escala*;
- Campos:
 - *esc_id*: identificador, tipo **int**
 - *esc_data_ini*: data e hora de início de escala, tipo **datetime**
 - *esc_data_fim*: data e hora de fim de escala, tipo **datetime**
 - *voo_id*: identificador do voo a que respeita a escala, tipo **int**
- Constraints:
 - Os quatro campos desta tabela não aceitam o valor *null*;
 - O campo *esc_id* é a chave primária;
 - O campo *voo_id* é chave estrangeira que relaciona a tabela *escala* com a tabela *voo*;
 - O campo *esc_data_fim* tem de ter uma data mais recente que a data do campo *esc_data_ini*.

EXEC sp_help 'funcionario';

Name	Owner	Type	Created_datetime
funcionario	dbo	user table	2019-06-10 10:53:04.333

Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource	Collation
func_id	int	no	4	10	0	no	(n/a)	(n/a)	NULL
func_nome	nvarchar	no	500			no	(n/a)	(n/a)	Latin1_General_CI_AS
func_tipo	nvarchar	no	200			no	(n/a)	(n/a)	Latin1_General_CI_AS
esc_id	int	no	4	10	0	yes	(n/a)	(n/a)	NULL

Identity	Seed	Increment	Not For Replication
func_id	1	1	0

RowGuidCol
No rowguidcol column defined.

Data_located_on_filegroup
PRIMARY

index_name	index_description	index_keys
funcionario_func_id_pk	clustered, unique, primary key located on PRIMARY	func_id

constraint_type	constraint_name	delete_action	update_action	status_enabled	status_for_replication	constraint_keys
FOREIGN KEY	funcionario_esc_id_fk	No Action	No Action	Enabled	Is_For_Replication	esc_id REFERENCES CrashJets.dbo.escala (esc_id)
PRIMARY KEY (clustered)	funcionario_func_id_pk	(n/a)	(n/a)	(n/a)	(n/a)	func_id
CHECK on column func_tipo	funcionario_func_tipo_ck	(n/a)	(n/a)	Enabled	Is_For_Replication	((func_tipo)='Apoio Administrativo' OR (func_tipo)='Assistente de Embarque' OR (func_tipo)='Tripulação')
CHECK on column func_id	funcionario_maximo_funcionarios_ck	(n/a)	(n/a)	Enabled	Is_For_Replication	((func_id)<=120)

- Tabela: *funcionario*;
- Campos:
 - *func_id*: identificador, tipo **int**
 - *func_nome*: nome do funcionário, tipo **nvarchar(250)**
 - *func_tipo*: tipo de funcionário de entre três possíveis tipos: 'Apoio Administrativo', 'Assistente de Embarque' e 'Tripulação', tipo **nvarchar(100)**
 - *esc_id*: identificador da próxima escala da qual o funcionário fará parte, tipo **int**
- Constraints:
 - Os campos desta tabela não aceitam o valor *null* exceto o campo *esc_id* (um funcionário pode não estar em determinado momento nalguma escala);

- O campo *func_id* é a chave primária;
- O campo *esc_id* é chave estrangeira que relaciona a tabela *escala* com a tabela *funcionário*;
- O campo *func_id* está limitado ao valor de 120 que é o número máximo de funcionários da empresa. Pode existir melhoria e controle de redundância ao converter este campo para chave externa, para uma tabela a ser criada para armazenar todos os tipos de funcionários - não implementado;
- O campo *func_tipo* tem de ter um de três valores possíveis: 'Apoio Administrativo', 'Assistente de Embarque' e 'Tripulação'.

EXEC sp_help 'reserva';

Name	Owner	Type	Created_datetime
reserva	dbo	user table	2019-06-11 00:35:37.190

Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource	Collation
res_id	int	no	4	10	0	no	(n/a)	(n/a)	NULL
bilhete_emitido	nchar	no	2			no	(n/a)	(n/a)	Latin1_General_CI_AS
res_data	datetime	no	8			no	(n/a)	(n/a)	NULL
voo_id	int	no	4	10	0	no	(n/a)	(n/a)	NULL
pass_id	int	no	4	10	0	no	(n/a)	(n/a)	NULL
ref_id	int	no	4	10	0	yes	(n/a)	(n/a)	NULL
preco_extras	int	no	4	10	0	no	(n/a)	(n/a)	NULL
preco_final	int	yes	4	10	0	yes	(n/a)	(n/a)	NULL

Identity	Seed	Increment	Not For Replication
res_id	1	1	0

RowGUIDCol
No rowguidcol column defined

Data located on filegroup
PRIMARY

Index_name	Index_description	Index_keys
reserva_res_id_pk	clustered, unique, primary key located on PRIMARY	res_id

constraint_type	constraint_name	delete_action	update_action	status_enabled	status_for_replication	constraint_keys
1	DEFAULT on column preco_extras	(n/a)	(n/a)	(n/a)	(n/a)	((0))
2	CHECK on column bilhete_emitido	(n/a)	(n/a)	Enabled	Is_For_Replication	((bilhete_emitido='N' OR bilhete_emitido='S'))
3	FOREIGN KEY	No Action	No Action	Enabled	Is_For_Replication	pass_id
4						REFERENCES CrashJets.dbo.passageiro (pass_id)
5	CHECK on column preco_extras	(n/a)	(n/a)	Enabled	Is_For_Replication	((preco_extras)>=0)
6	FOREIGN KEY	No Action	No Action	Enabled	Is_For_Replication	ref_id
7						REFERENCES CrashJets.dbo.refeicao (ref_id)
8	CHECK Table Level	(n/a)	(n/a)	Enabled	Is_For_Replication	((dbo.[uf_IsValidResData])=[res_data].[voo_id]=1))
9	PRIMARY KEY (clustered)	(n/a)	(n/a)	(n/a)	(n/a)	res_id
10	CHECK Table Level	(n/a)	(n/a)	Enabled	Is_For_Replication	((dbo.[uf_IsVooDurMajor2])=[voo_id]<=1) AND ([ref_id] IS NULL OR [dbo].[uf_IsVooDurMajor2])=[voo_id]=1) AND ([ref_id]=3) OR ([ref_id]=2) OR ([ref_id]=1))
11	FOREIGN KEY	No Action	No Action	Enabled	Is_For_Replication	voo_id
12						REFERENCES CrashJets.dbo.voo (voo_id)

- Tabela: *reserva*;
- Campos:
 - *res_id*: identificador, tipo **int**
 - *bilhete_emitido*: indicador de que o preço final da reserva se encontra devidamente pago e o bilhete foi emitido (tem dois valores possíveis 'N' ou 'S'), tipo **nchar(1)**
 - *res_data*: data e hora de realização da reserva, tipo **datetime**
 - *voo_id*: identificador do voo a que respeita a reserva, tipo **int**
 - *pass_id*: identificador do passageiro que efetuou a reserva, tipo **int**
 - *ref_id*: identificador da refeição, tipo **int**
 - *preco_extras*: valor de extras (por exemplo, bagagem) que a companhia necessita cobrar ao cliente (este valor soma ao preço base do voo), tipo **int**
 - *preco_final*: campo calculado a partir da função *udf_GetPrecoFinal* que recebe como parâmetros o *voo_id*, o *pass_id* e o *preco_extras* e calcula o preço final da reserva segundo a seguinte fórmula:

$$preco_final = (preco_base + preco_extras) * ((100 - percentagem_desconto) / 100)$$

- Constraints:
 - Todos os campos desta tabela não aceitam o valor *null* exceto os campos *ref_id* e *preco_final* (no caso do campo calculado *preco_final* deveria ter uma *constraint not null* e um *default 0* contudo não sendo uma coluna regular, só pode ter *constraints primary key* e *unique*. Para uma implementação mais correta, deveríamos ter criado uma *view* em vez de uma função chamada pelo campo calculado, a fim de evitarmos ter um campo que aceita valores *null* na tabela *reserva*. Ainda assim, é utilizada a função *udf_GetPrecoFinal* que garante inserir um valor inteiro válido no campo *preco_final*.

Devem ser limitados através das permissões de utilizador as inserções de valores *null* na coluna *preco_final* como alternativa);

- O campo *res_id* é a chave primária;
- O campo *pass_id* é chave estrangeira que relaciona a tabela *passageiro* com a tabela *reserva*;
- O campo *voo_id* é chave estrangeira que relaciona a tabela *voo* com a tabela *reserva*;
- O campo *ref_id* é chave estrangeira que relaciona a tabela *refeicao* com a tabela *reserva*;
- O campo *bilhete_emitido* tem de ter um de dois valores: 'S' ou 'N';
- O campo *preco_extras* tem de ter um valor não negativo;
- A reserva deve ser feita numa data anterior à data do voo (esta verificação é realizada através da função *udf_IsValidResData* que recebe como parâmetros o *voo_id* e a *res_data*);
- O passageiro tem direito a refeição quando o voo tem duração superior a duas horas pelo que *ref_id* é *null* quando o voo tem duração não superior a duas horas, caso contrário terá um de três valores válidos: '1', '2' ou '3' (a verificação da duração do voo é realizada através da função *udf_IsVooDurMaior2H* que recebe como parâmetro o *voo_id*).

1.3.5. User Defined Functions

Identificam-se em seguida as funções criadas para a correta implementação da solução CrashJets:

1.3.5.1. Udf_GetPrecoFinal

```
EXEC sp_help 'udf_GetPrecoFinal';
```

	Name	Owner	Type	Created_datetime
1	udf_GetPrecoFinal	dbo	scalar function	2019-06-11 00:35:37.053

	Parameter_name	Type	Length	Prec	Scale	Param_order	Collation
1		int	4	10	0	0	NULL
2	@VooId	int	4	10	0	1	NULL
3	@PrecoExtras	int	4	10	0	2	NULL
4	@PassId	int	4	10	0	3	NULL

A função **udf_GetPrecoFinal** recebe como parâmetros o identificador do voo, o valor do preço dos extras cobrados ao cliente (por exemplo, de bagagem) e o identificador do passageiro. Devolve um valor inteiro indicando o preço final de custo da reserva.

O cálculo efetuado verifica quantas milhas tem o cartão de milhas do passageiro para definir qual a percentagem de desconto a aplicar caso haja direito a desconto (o cartão de milhas do passageiro tenha pelo menos 1000 milhas efetuadas na companhia).

O preço final é resultado da subtração do desconto com a soma do preço base do voo com o preço dos extras cobrados.

Fórmula:

$$pr_fin = (pr_b + extr) * ((100 - perc_desc) / 100)$$

1.3.5.2. Udf_IsValidResData

```
EXEC sp_help 'udf_IsValidResData';
```

	Name	Owner	Type	Created_datetime
1	udf_IsValidResData	dbo	scalar function	2019-06-11 00:35:37.103

	Parameter_name	Type	Length	Prec	Scale	Param_order	Collation
1		bit	1	1	NULL	0	NULL
2	@ResData	datetime	8	23	3	1	NULL
3	@ResVoold	int	4	10	0	2	NULL

A função **udf_IsValidResData** recebe como parâmetros a data de realização da reserva e o identificador do voo. Devolve um valor de tipo *bit* indicando se a data de realização da reserva é anterior à data de realização do voo. Não se podem efetuar reservas para um voo após a sua data e hora de realização.

1.3.5.3. Udf_IsVooDurMaior2H

```
EXEC sp_help 'udf_IsVooDurMaior2H';
```

	Name	Owner	Type	Created_datetime
1	udf_IsVooDurMaior2H	dbo	scalar function	2019-06-11 00:35:37.107

	Parameter_name	Type	Length	Prec	Scale	Param_order	Collation
1		bit	1	1	NULL	0	NULL
2	@ResVoold	int	4	10	0	1	NULL

A função **udf_IsVooDurMaior2H** recebe como parâmetro o identificador do voo. Devolve um valor de tipo *bit* indicando se a duração do voo ultrapassa as duas horas. Esta função permite definir, por exemplo, se o passageiro terá direito a refeição durante o voo ou não.

1.3.5.4. Udf_AviaoEstaDisponivel

```
EXEC sp_help 'udf_AviaoEstaDisponivel';
```

	Name	Owner	Type	Created_datetime
1	udf_AviaoEstaDisponivel	dbo	scalar function	2019-06-11 00:35:37.110

	Parameter_name	Type	Length	Prec	Scale	Param_order	Collation
1		bit	1	1	NULL	0	NULL
2	@Aviaold	int	4	10	0	1	NULL
3	@Voold	int	4	10	0	2	NULL

A função **udf_AviaoEstaDisponivel** recebe como parâmetros o identificador do voo e o identificador do avião. Devolve um valor de tipo *bit* indicando se a data de manutenção do avião coincide ou não com o intervalo temporal definido pelas datas de partida e de chegada do voo. Um voo que se realize num determinado intervalo temporal só pode ser realizado por um avião que não esteja em manutenção algures durante esse intervalo temporal. Se a data de manutenção do avião for anterior à data de partida do voo ou posterior à data de chegada do voo, o avião pode ser utilizado para a realização do voo.

1.3.6. User Stored Procedures

Identificam-se em seguida os procedimentos criados para a correta implementação da solução CrashJets:

1.3.6.1. Usp_ListaPassageiros

```
EXEC sp_help 'usp_ListaPassageiros';
```

	Name	Owner	Type	Created_datetime
1	usp_ListaPassageiros	dbo	stored procedure	2019-06-11 00:35:37.117

	Parameter_name	Type	Length	Prec	Scale	Param_order	Collation
1	@Voold	int	4	10	0	1	NULL

O procedimento **usp_ListaPassageiros** recebe como parâmetro o identificador do voo. Permite obter a listagem completa de passageiros do voo (inclui passageiros e funcionários da companhia escalados para esse voo). Mostra o identificador do passageiro/funcionário, o nome do passageiro/funcionário, a data do voo e os locais de partida e de chegada do voo.

1.3.6.2. Usp_EfetuarReserva

```
EXEC sp_help 'usp_EfetuarReserva';
```

	Name	Owner	Type	Created_datetime
1	usp_EfetuarReserva	dbo	stored procedure	2019-06-11 00:35:37.123

	Parameter_name	Type	Length	Prec	Scale	Param_order	Collation
1	@Voold	int	4	10	0	1	NULL
2	@PassId	int	4	10	0	2	NULL
3	@RefId	int	4	10	0	3	NULL
4	@PrecoExtras	int	4	10	0	4	NULL

O procedimento **usp_EfetuarReserva** recebe como parâmetros o identificador do voo, o identificador do passageiro, o identificador da refeição (que pode não ser necessário caso o voo não exceda as duas horas de duração) e o preço dos extras. Permite efetuar uma reserva para um voo com validação de lotação do avião.

A verificação começa por comparar a lotação do avião que realizará o voo com o número de reservas já realizadas (devidamente pagas pelo cliente com bilhete emitido) para o voo.

Se não houver mais lugares disponíveis para o voo é mostrada a informação de '*Overbooking*' e a reserva não pode ser e não é efetuada por não ser permitido *overbooking*.

Caso haja lugares ainda disponíveis, a reserva é efetuada. Verifica-se se a duração do voo é superior a duas horas para definir se o passageiro tem direito a refeição. A reserva é feita com indicação de bilhete emitido, para o voo com *voo_id*, para o passageiro com o *pass_id*, com ou sem refeição consoante a duração do voo e com o preço final calculado de acordo com o campo calculado *preco_final* da tabela *reserva* (através da função **udf_GetPrecoFinal**).

No final é atualizado o valor de milhas do cartão de milhas do passageiro, ao qual é somado o valor da distância em milhas do voo que o passageiro acabou de reservar com sucesso.

1.3.6.3. Usp_Lotacao

```
EXEC sp_help 'usp_Lotacao';
```

	Name	Owner	Type	Created_datetime
1	usp_Lotacao	dbo	stored procedure	2019-06-11 00:35:37.130

	Parameter_name	Type	Length	Prec	Scale	Param_order	Collation
1	@Voold	int	4	10	0	1	NULL

O procedimento **usp_Lotacao** recebe como parâmetro o identificador do voo. Permite obter o horário e lotação de determinado voo.

A verificação começa por comparar a lotação do avião que realizará o voo com o número de reservas já realizadas (devidamente pagas pelo cliente com bilhete emitido) para o voo.

Se não houver mais lugares disponíveis para o voo é mostrada a informação de '*Lotação esgotada*', o identificador do voo, a data do voo e os locais de partida e de chegada do voo.

Se houver lugares disponíveis para o voo é mostrada a informação de quantas vagas ainda existem disponíveis para o voo, o identificador do voo, a data do voo e os locais de partida e de chegada do voo.

1.4. Política de *Backups* e *Disaster Recovery*

1.4.1. Política de *Backups*

Resumo da política adotada:

- **Periodicidade:** Diário
- **Hora:** 03:00
- **Localização BackupDev:** C:\BACKUP\BackupDev.bak
- **Localização MirrorDev:** C:\BACKUP\MirrorDev.bak
- **Modelo de recuperação:** *database full*
- **Execução:** Membro *sysadmin Server Role*, *db_owner* e *bd_backupoperator DB Roles*

Durante o processo de *backup* não é possível criar ou modificar a base de dados, executar operações na BD, criar índices, executar operações de *Shrinking* ou *Nonlogged*.

Tendo em conta os requisitos de operação para o projeto foi definido o tipo de *backup* como sendo *Database* e o modo de recuperação como sendo *Full*.

Foram criados na diretoria **C:\BACKUP** dois *backup devices* denominados **BackupDev** (ficheiro C:\BACKUP\BackupDev.bak) e **MirrorDev** (ficheiro C:\BACKUP\MirrorDev.bak).

```
exec sp_helpdevice;
```

	device_name	physical_name	description	status	ontrtype	size
1	BackupDev	C:\BACKUP\BackupDev.bak	disk, backup device	16	2	0
2	MirrorDev	C:\BACKUP\MirrorDev.bak	disk, backup device	16	2	0

O processo de *backup* manual da base de dados *CrashJets* realiza-se utilizando estes dois *backup devices*.

A *query* T-SQL seguinte permite efetuar o **backup** da base de dados para os dois *backup devices* sendo um o *backup* principal e o outro um *backup* espelho:

```
USE master
```

```
backup database CrashJets to BackupDev
```

```
mirror to MirrorDev
```

```
with format, checksum
```

O resultado da *query* acima indicada é o seguinte:

```
Processed 512 pages for database 'CrashJets', file 'CrashJets_Primary' on file 1.
Processed 24 pages for database 'CrashJets', file 'C:\CrashJets\CrashJets_Secondary01'
on file 1.
Processed 2 pages for database 'CrashJets', file 'CrashJets_Log' on file 1.
BACKUP DATABASE successfully processed 538 pages in 1.160 seconds (3.623 MB/sec).
```

A verificação da validade do *backup* efetuado é realizada através dos seguintes comandos:

```
restore verifyonly from BackupDev with checksum;
```

```
restore verifyonly from MirrorDev with checksum;
```

O resultado dos comandos executados é o seguinte:

```
The backup set on file 1 is valid.
```

```
The backup set on file 1 is valid.
```

De forma a automatizar a tarefa, foi criado via interface do SSMS e testado um *job* para executar o *backup* para uma frequência diária, às 03:00h (por ser uma hora em que se estima menor carga/utilização da base de dados). Este *job* também pode ser executado manualmente sempre que o *Database Administrator* assim o considere necessário.

1.4.2. Política de *Disaster Recovery*

No caso de haver a necessidade de recuperação da base de dados (por exemplo, caso o *primary data file* **CrashJets_Principal.mdf** fique corrompido ou inacessível), podem ser executados manualmente os seguintes comandos para *restore* do último *backup* realizado.

```
-- RESTORE DB
-- take off line
USE [master]
GO
ALTER DATABASE [CrashJets] SET OFFLINE
GO

-- restore DB a partir do backup principal
RESTORE DATABASE [CrashJets] FROM BackupDev WITH REPLACE
--ou restore DB a partir do backup de espelho
RESTORE DATABASE [CrashJets] FROM MirrorDev WITH REPLACE

-- take on line
USE [master]
GO
ALTER DATABASE [CrashJets] SET ONLINE
GO
```

O tipo de *backup* efetuado é *Database (full)* para a mesma base de dados.

De forma a automatizar a tarefa, foram criados via interface do SSMS e testados dois *jobs*: um para executar *backup* e outro para executar *restore*. Também podem ser executados manualmente sempre que o *Database Administrator* assim o considere necessário.

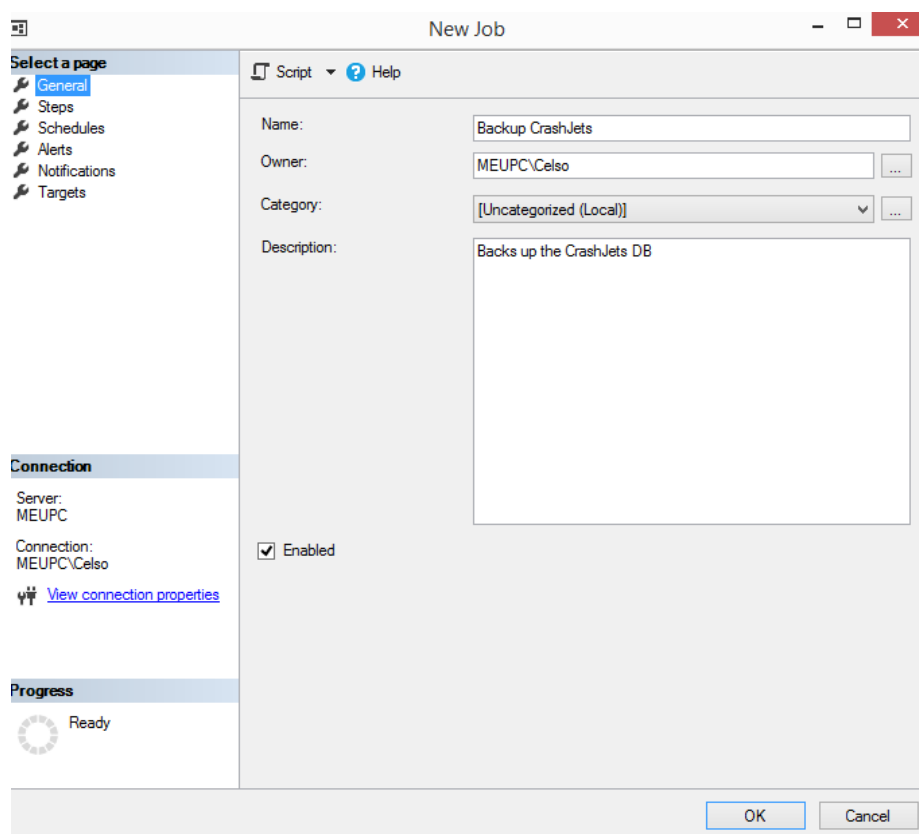


Figura 4 – Criação de *job* para automatizar *backup*

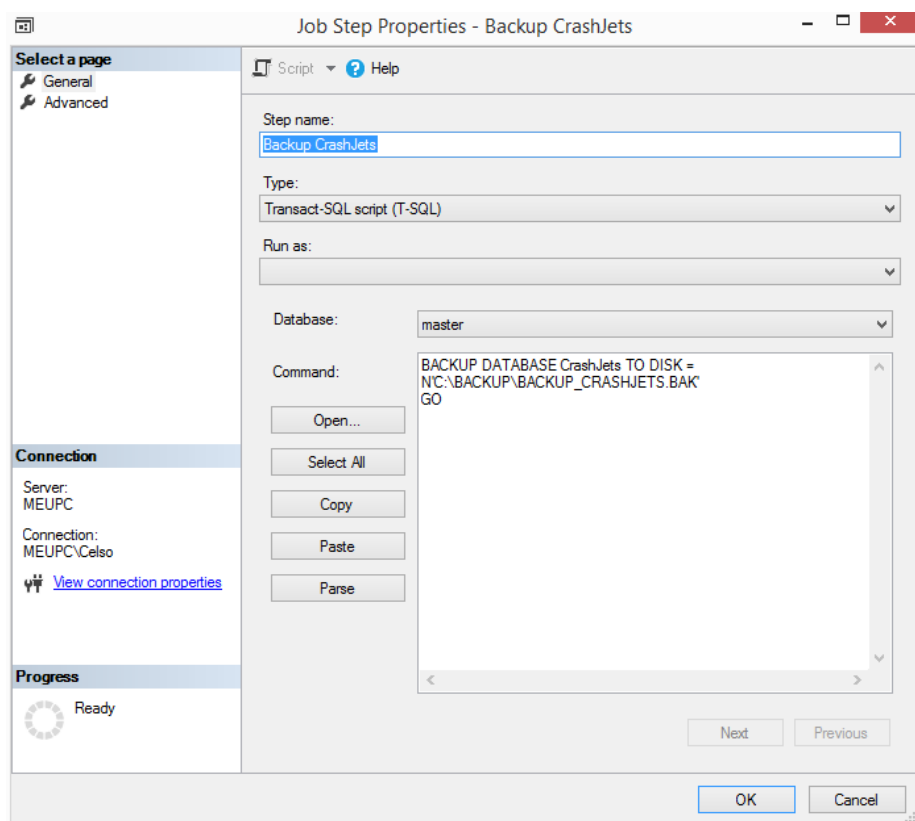


Figura 5 – Criação de *job* para automatizar *backup* (continuação)

New Job Schedule

Name: Jobs in Schedule

Schedule type: Recurring ☒ Enabled

One-time occurrence

Date: 12/06/2019 Time: 02:22:36

Frequency

Occurs: Daily

Recurs every: 1 day(s)

Daily frequency

☒ Occurs once at: 03:00:00

☐ Occurs every: 1 hour(s)

Starting at: 00:00:00

Ending at: 23:59:59

Duration

Start date: 12/06/2019 ☐ End date: 12/06/2019

☒ No end date:

Summary

Description: Occurs every day at 03:00:00. Schedule will be used starting on 12/06/2019.

OK Cancel Help

Figura 6 – Criação de *job* para automatizar *backup* (continuação)

New Job

Select a page

- General
- Steps
- Schedules
- Alerts
- Notifications
- Targets

Script ? Help

Actions to perform when the job completes:

☒ E-mail: CrashJetsOperator When the job fails

☐ Page: When the job fails

☐ Write to the Windows Application event log: When the job fails

☐ Automatically delete job: When the job succeeds

Connection

Server: MEUPC

Connection: MEUPC\Celso

[View connection properties](#)

Progress

Ready

OK Cancel

Figura 7 – Criação de *job* para automatizar *backup* (continuação)

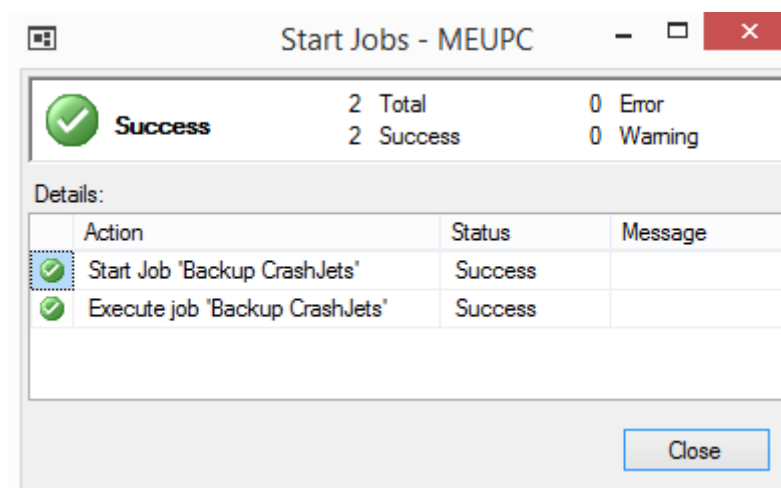


Figura 8 – Execução de *job* para *backup*

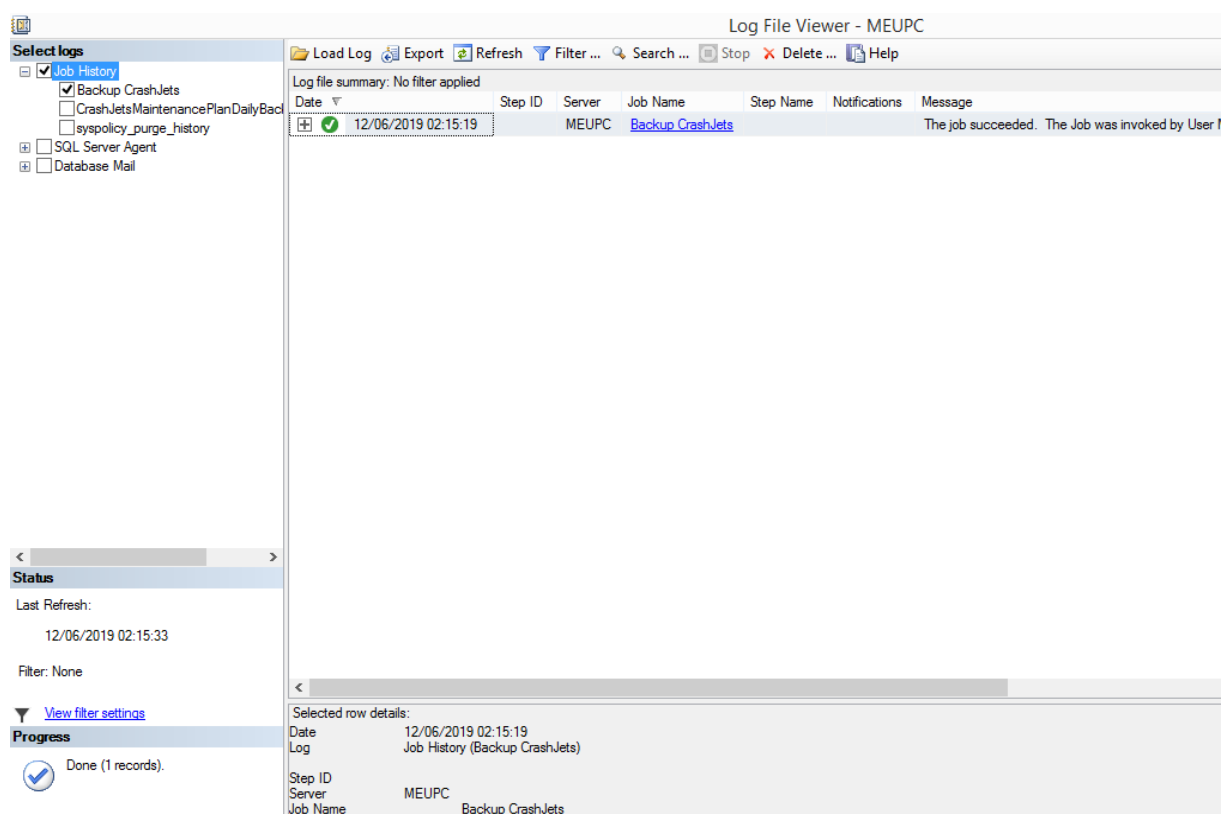


Figura 9 – Job Log File Viewer

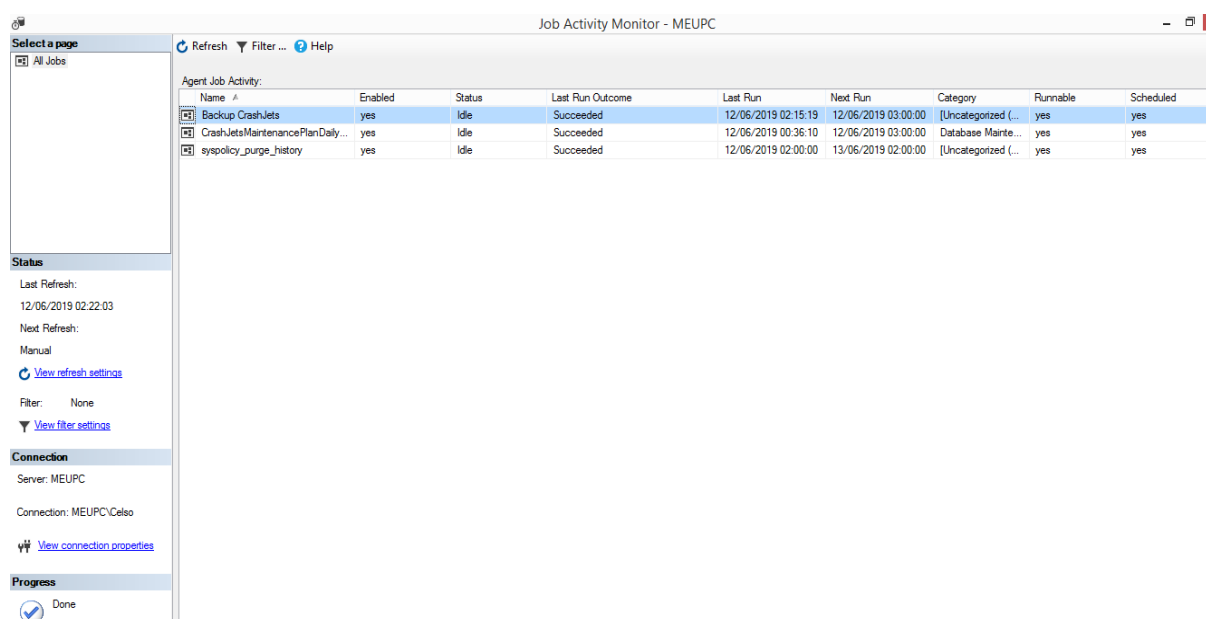


Figura 10 – Job Activity Monitor

```
select name,description from msdb.dbo.sysjobs;
```

	name	description
1	CrashJetsMaintenancePlanDailyBackup.Subplan_1	No description available.
2	Backup CrashJets	Backs up the CrashJets DB
3	syspolicy_purge_history	No description available.

A automatização permite implementar uma política de gestão proactiva dando ênfase às tarefas vitais de manutenção e diminuindo a quantidade de trabalho administrativo e a probabilidade de erro humano.

1.4.3. Plano de manutenção

De forma a automatizar a realização de *backups* com geração do respetivo relatório de *backup* em formato texto, foi criado via interface do SSMS e testado um plano de manutenção para executar um *backup* automático com uma frequência diária, às 03:00h, por ser uma hora em que se estima menor carga/utilização da base de dados. Este plano de manutenção também pode ser executado manualmente sempre que o *Database Administrator* assim o considere necessário.

New Job Schedule

Name: Jobs in Schedule

Schedule type: ☒ Enabled

One-time occurrence

Date: Time:

Frequency

Occurs:

Recurs every: day(s)

Daily frequency

☒ Occurs once at:

☐ Occurs every: hour(s) Starting at: Ending at:

Duration

Start date: ☐ End date: ☒ No end date:

Summary

Description:

OK Cancel Help

Figura 11 – Criação do Plano de Manutenção

Maintenance Plan Wizard

Complete the Wizard
Verify the choices made in the wizard, and then click Finish.

Click Finish to perform the following actions:

- Maintenance Plan Wizard
 - Create Maintenance Plan 'CrashJetsMaintenancePlanDailyBackup'
 - Description: CrashJets Maintenance Plan Daily Backup
 - Single schedule selected
 - SQL Server Agent job is scheduled to run : Occurs every day at 03:00:00. Schedule will be used starting on
 - Define Back Up Database (Full) Task
 - Backup Database on
 - Databases: CrashJets
 - Type: Full
 - Append existing
 - Destination: Disk
 - Backup Compression (Default)
 - Selected reporting options
 - Report will be generated in folder: C:\BACKUP

Help < Back Next > Finish Cancel

Figura 12 – Criação do Plano de Manutenção (continuação)

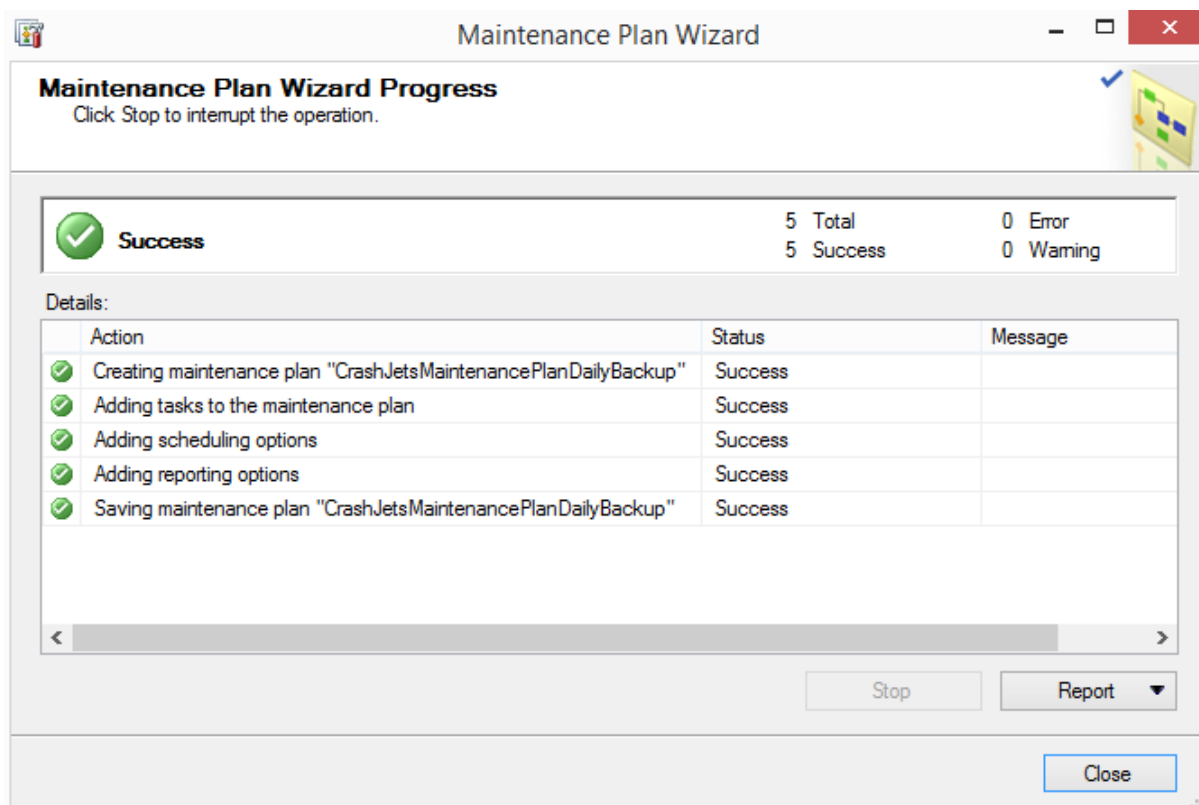


Figura 13 – Criação do Plano de Manutenção (continuação)

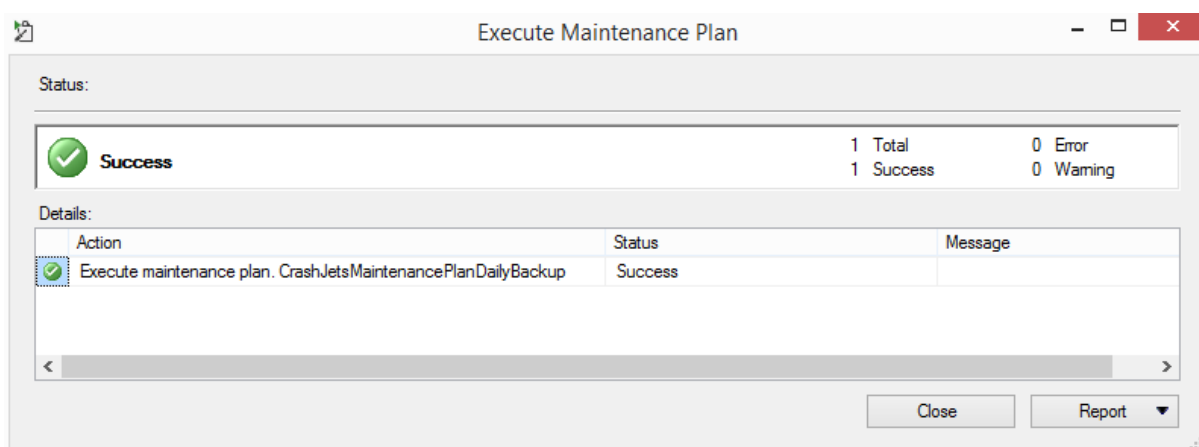


Figura 14 – Execução do Plano de Manutenção

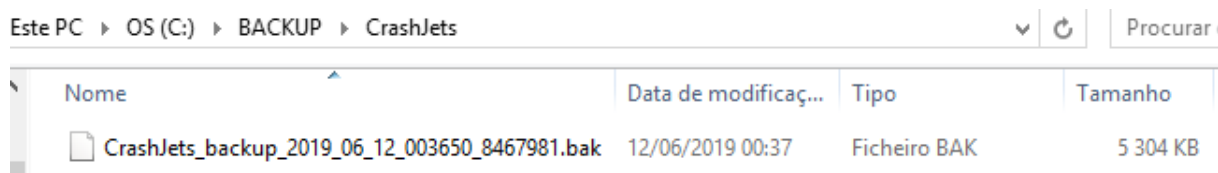


Figura 15 – Ficheiro de *backup* criado

Este PC ▸ OS (C:) ▸ BACKUP ▸

Nome	Data de modificaç...	Tipo
CrashJets	12/06/2019 00:36	Pasta de ficheiros
BackupDev.bak	07/06/2019 16:46	Ficheiro BAK
CrashJetsMaintenancePlanDailyBackup_Subplan_1_20190612003712.txt	12/06/2019 00:37	Documento de tex...
MirrorDev.bak	07/06/2019 16:46	Ficheiro BAK

Figura 16 – Relatório de *backup* gerado

```
Microsoft(R) Server Maintenance Utility (Unicode) Version
14.0.2014
Report was generated on "MEUPC".
Maintenance Plan: CrashJetsMaintenancePlanDailyBackup
Duration: 00:00:34
Status: Succeeded.
Details:
Back Up Database (Full) (MEUPC)
Backup Database on Local server connection
Databases: CrashJets
Type: Full
Append existing
Task start: 2019-06-12T00:36:52.
Task end: 2019-06-12T00:37:11.
Success
Command:EXECUTE master.dbo.xp_create_subdir N'C:\BACKUP
\CrashJets'

GO
BACKUP DATABASE [CrashJets] TO DISK = N'C:\BACKUP\CrashJets
\CrashJets_backup_2019_06_12_003650_8467981.bak' WITH
NOFORMAT, NOINIT, NAME = N'CrashJets_backup_2019_06_12_
003650_8467981'', SKIP, REWIND, NOUNLOAD, STATS = 10

GO
```

Figura 17 – Relatório de *backup* gerado (conteúdo)

De forma a monitorizar de forma mais prática o tamanho do *log file*, foi criado via interface do SSMS um operador e um alerta indicativo da percentagem de ocupação do mesmo ter chegado a 80% do seu tamanho máximo. O operador/Database Administrator receberá na sua conta de *email* o alerta e poderá agir em tempo útil.

New Alert

Select a page: General, Response, Options

Script Help

Name: Log File Size Alert ☒ Enable

Type: SQL Server performance condition alert

Performance condition alert definition

Object: Databases

Counter: Percent Log Used

Instance: CrashJets

Alert if counter: rises above Value: 80

Connection: Server: MEUPC, Connection: MEUPC\Celso, [View connection properties](#)

Progress: Ready

OK Cancel

Figura 18 – Criação de alerta para controlo do tamanho do *log file*

New Alert

Select a page: General, Response, Options

Script Help

☐ Execute job

Operator list:

Operator	Email	Pager
CrashJetsOperator	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Connection: Server: MEUPC, Connection: MEUPC\Celso, [View connection properties](#)

Progress: Ready

New Operator View Operator

OK Cancel

Figura 19 – Criação de alerta para controlo do tamanho do *log file* (continuação)

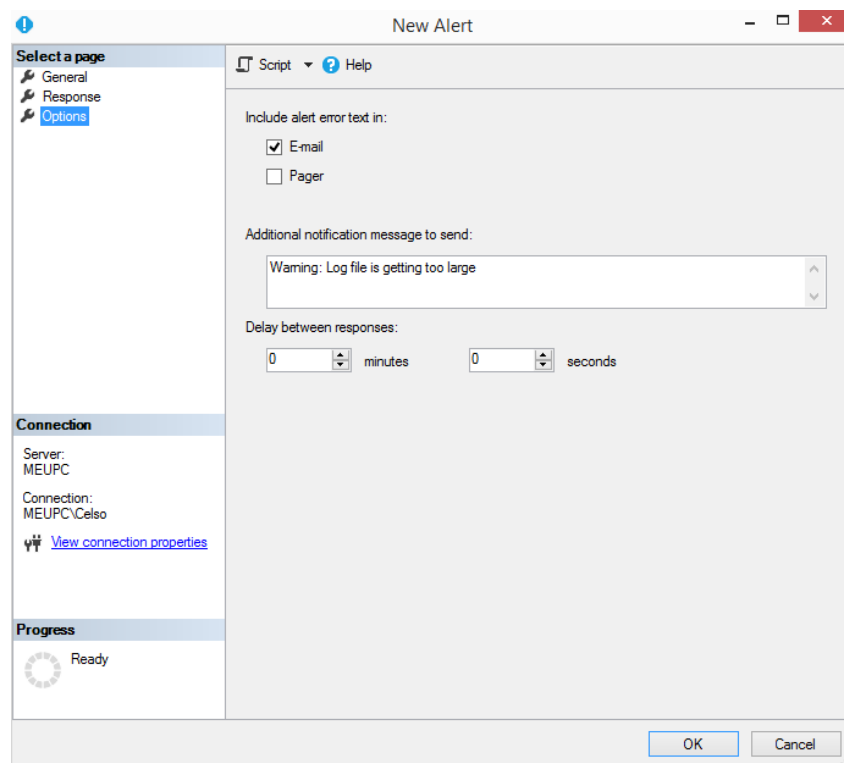


Figura 20 – Criação de alerta para controlo do tamanho do *log file* (continuação)

```
select * from sys.dm_db_log_space_usage;
```

	database_id	total_log_size_in_bytes	used_log_space_in_bytes	used_log_space_in_percent	log_space_in_bytes_since_last_backup
1	1	2088960	626688	30	131072

De forma a monitorizar de forma mais prática o tamanho do *data file*, foi criado via interface do SSMS um operador e um alerta indicativo do tamanho do mesmo ter chegado a 200MB (cerca de 80% do tamanho máximo definido). O operador/*Database Administrator* receberá na sua conta de *email* o alerta e poderá agir em tempo útil.

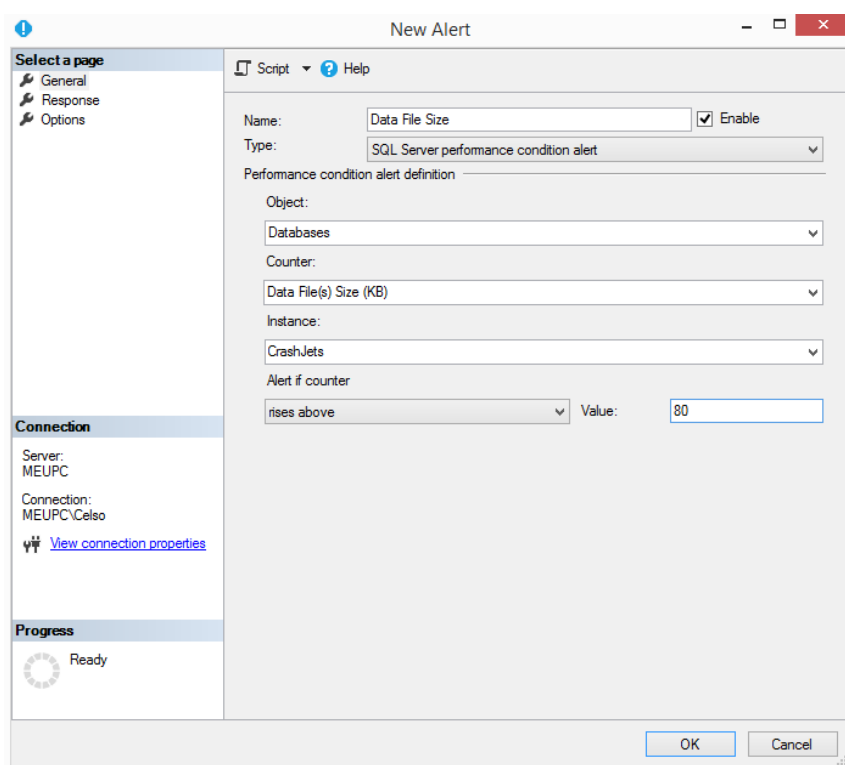


Figura 21 – Criação de alerta para controlo do tamanho do *data file*

1.4.4. Gestão de *Backups* e *Restores*

Tratando-se de uma etapa muito importante para a recuperação da BD é necessário realizar cópias frequentes de um local para o outro. O processo deverá ser realizado de forma rápida, para garantir a preservação dos dados e reduzindo as possibilidades de perda de informação importante e fundamental para o negócio.

Embora seja um processo automático há a necessidade de realizar manutenções periódicas e testes regulares de recuperação de informação, de forma a garantir a eficiência de todo o processo.

É necessário realizar um planeamento de *backup/restore* para validar se o *backup* é concluído com sucesso. Por exemplo, através de rotinas de testes de recuperação. A monitorização de forma constante e adequada contribuirá para a preservação dos dados nos mais diferentes cenários (cheias, incêndios, explosões, falhas de *backup*, roubo, etc).

Também a realização de *backups* para múltiplos *physical devices* determinam a velocidade do processo de *backup*, minimizando a atividade concorrente.

1.5. Performance Tunning

Foram definidos os seguintes onze *counters* para *trace* da *performance* do serviço SQL Server:

Processador/% de tempo do processador/(_Total): Permite monitorizar a percentagem de ocupação do CPU do servidor onde corre o serviço SQL Server. Uma taxa de ocupação de até 40% considera-se um valor aceitável;

Memória/Mbytes disponíveis: Permite monitorizar a distribuição equilibrada e necessária da memória disponível pelo SQL Server e pelos outros processos em execução no sistema operativo;

Ficheiro de paginação/% de utilização/(_Total): Permite controlar a memória virtual disponível de forma a evitar escritas para disco que diminuem a *performance* do sistema;

Disco físico/Média de disco seg. escritas/(_Total) e Disco físico/Média de disco seg. leituras/(_Total): Permitem controlar a taxa de resposta do sistema aos pedidos de informação ao disco (latência). Até 20ms considera-se um valor aceitável;

Interface de rede/Total de bytes seg.: Permite controlar a taxa de processamento de informação do adaptador de rede. Até 60% da largura de banda da rede considera-se um valor aceitável;

SQLServer:SQL Statistics/Batch Requests seg.: Permite controlar o número de pedidos de utilizadores ao SQL Server e a sua variância ao longo do tempo;

SQLServer:Databases/Active Transactions/CrashJets: Permite controlar o número de transações ativas. Quanto menos transações ativas estiverem durante o menor tempo possível a fazer *lock* a informação que estão a alterar/consultar, maior a efetividade da base de dados. Permite também detetar transações que estejam a bloquear informação sem nunca a libertarem para ser alterada/consultada por outras transações;

SQLServer:Locks/Lock requests seg./(_Total): Permite monitorizar a quantidade de recursos *lockados* para escrita/leitura de forma a aumentar a concorrência e a *performance*;

SQLServer:Buffer Manager/Page reads seg. e SQLServer:Buffer Manager/Page writes seg.: Permitem monitorizar o número de leituras e escritas de/para disco e atuar na melhoria de *performance* através da utilização de índices, aperfeiçoamento da arquitetura da base de dados, criação de *queries* mais eficientes, etc.

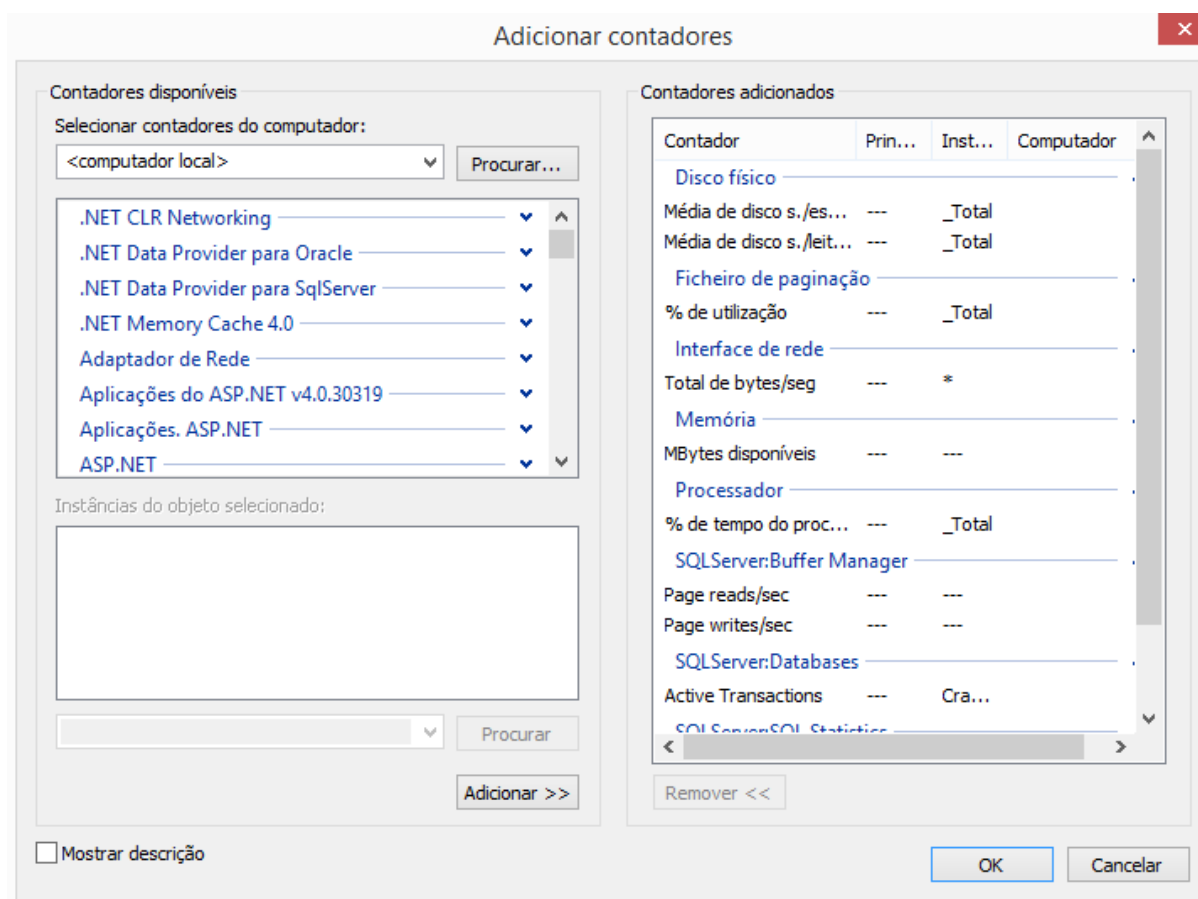


Figura 22 – Monitor de desempenho (*counters*)

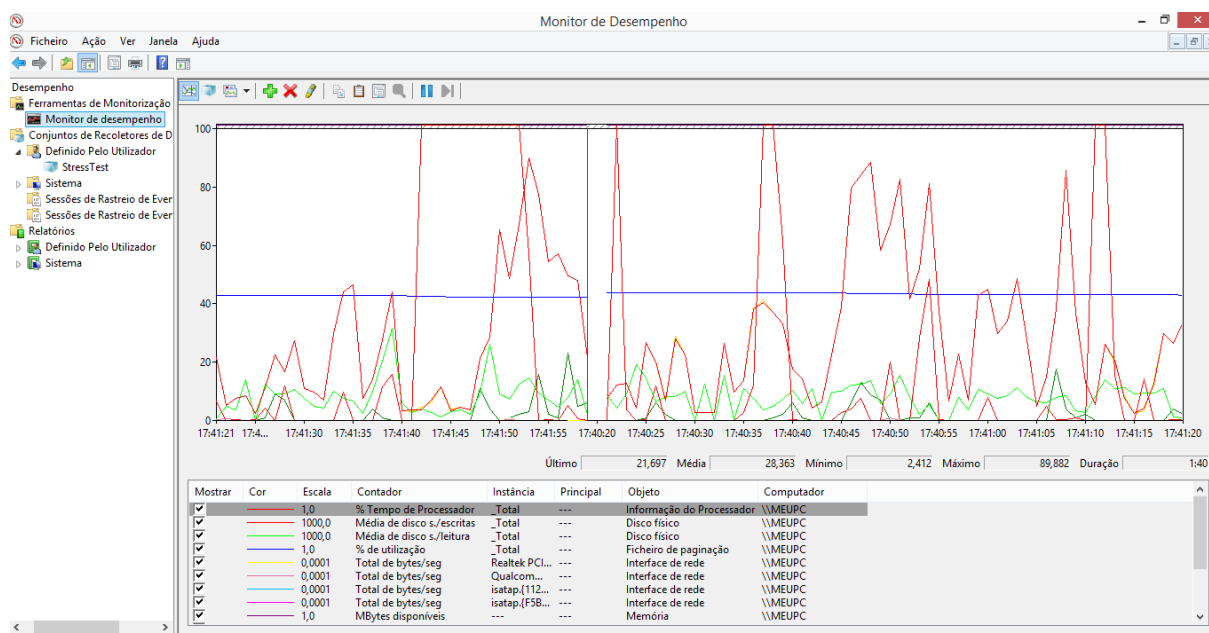


Figura 23 – Monitor de desempenho (gráfico após execução da *stored procedure usp_ListaPassageiros*)

Demonstra-se em seguida a definição de um *Stress Test* com os *counters* indicados para a base de dados CrashJets:

Criar novo Conjunto de Recolectores de Dados.

Como gostaria de criar este novo conjunto de recolectores de dados?

Nome:

☐ Criar a partir de um modelo (Recomendado)

☒ Criar manualmente (Avançado)

Figura 24 – Criação de *Stress Test*

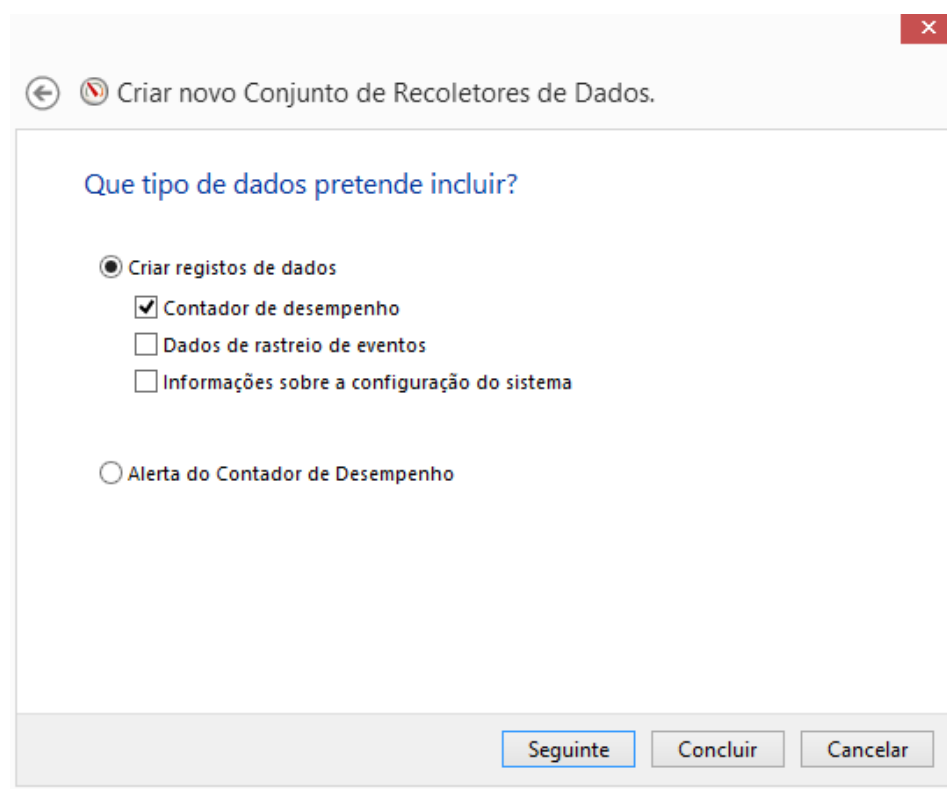


Figura 25 – Criação de *Stress Test* (continuação)

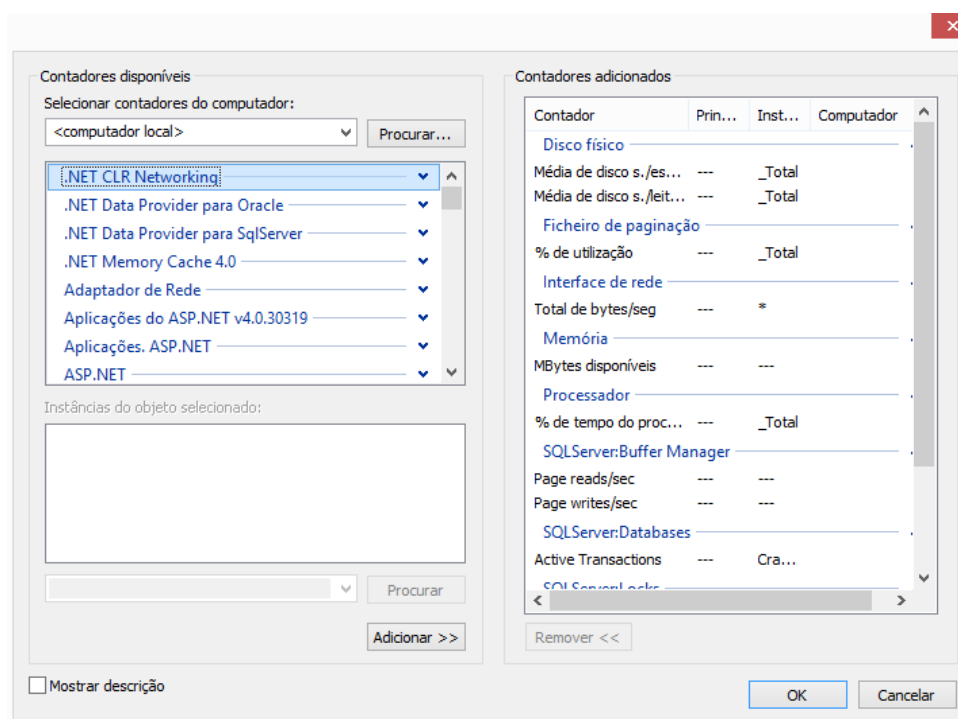


Figura 26 – Criação de *Stress Test* (counters)

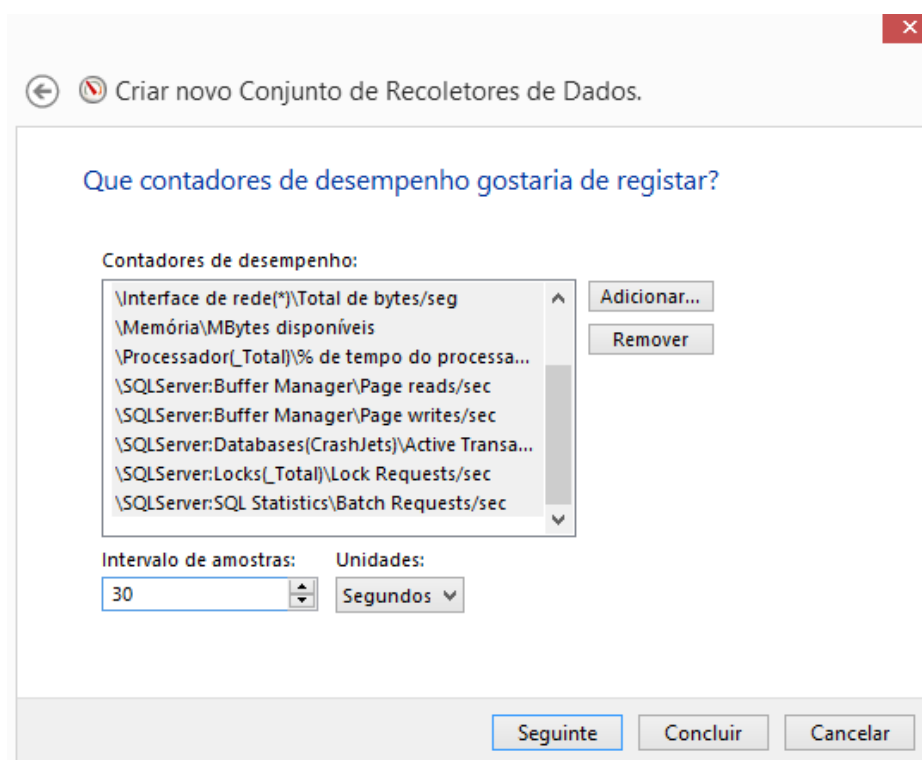


Figura 27 – Criação de *Stress Test* (continuação)

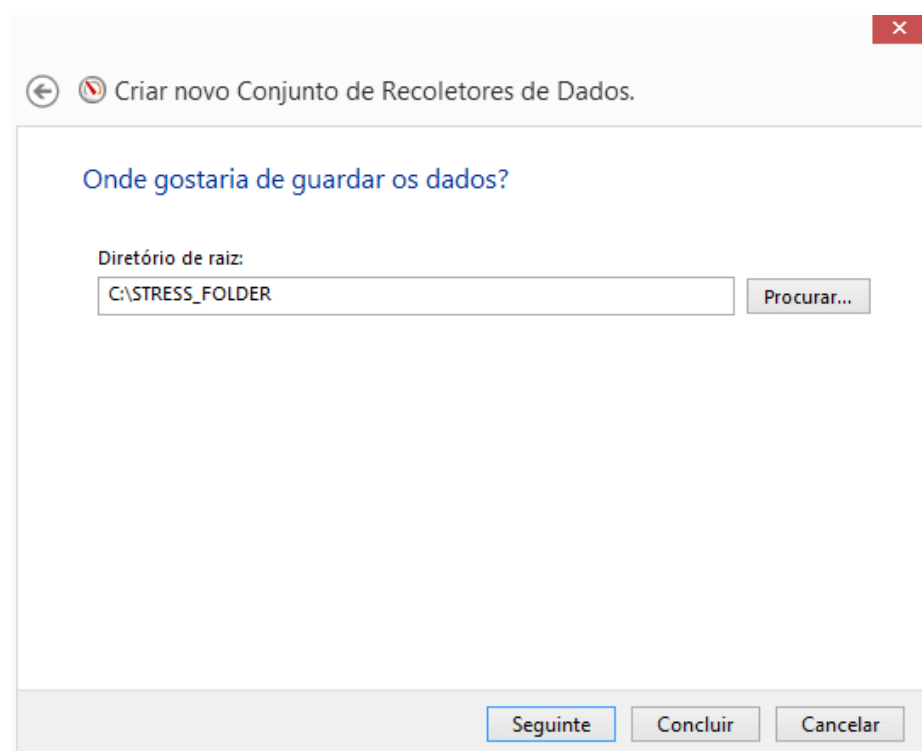


Figura 28 – Criação de *Stress Test* (continuação)

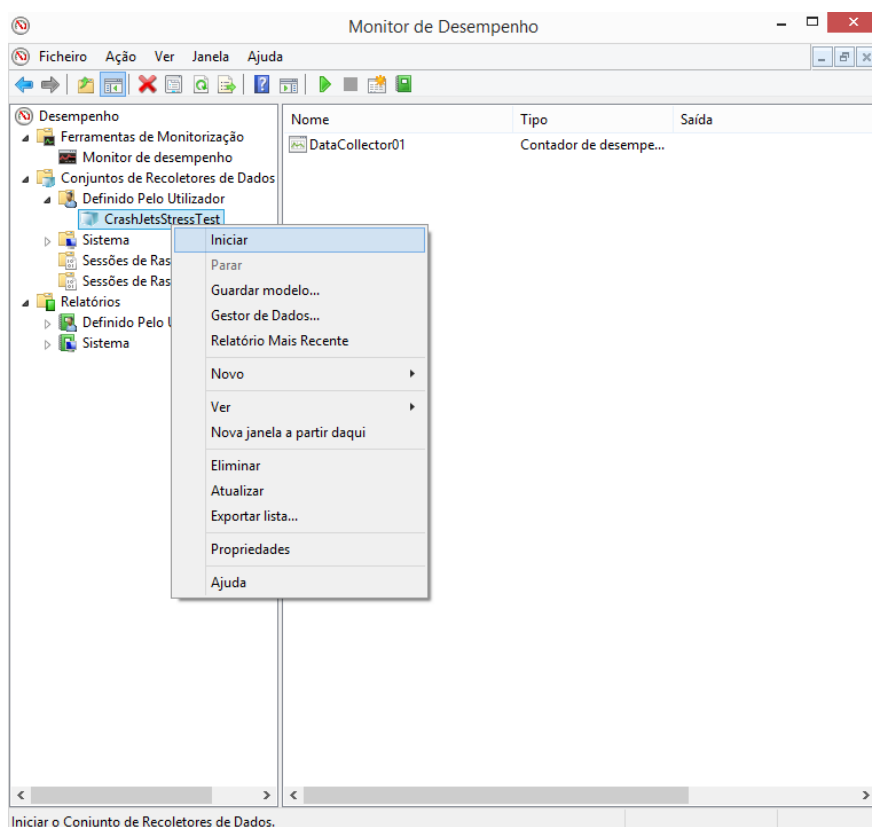


Figura 29 – Execução do *Stress Test* criado

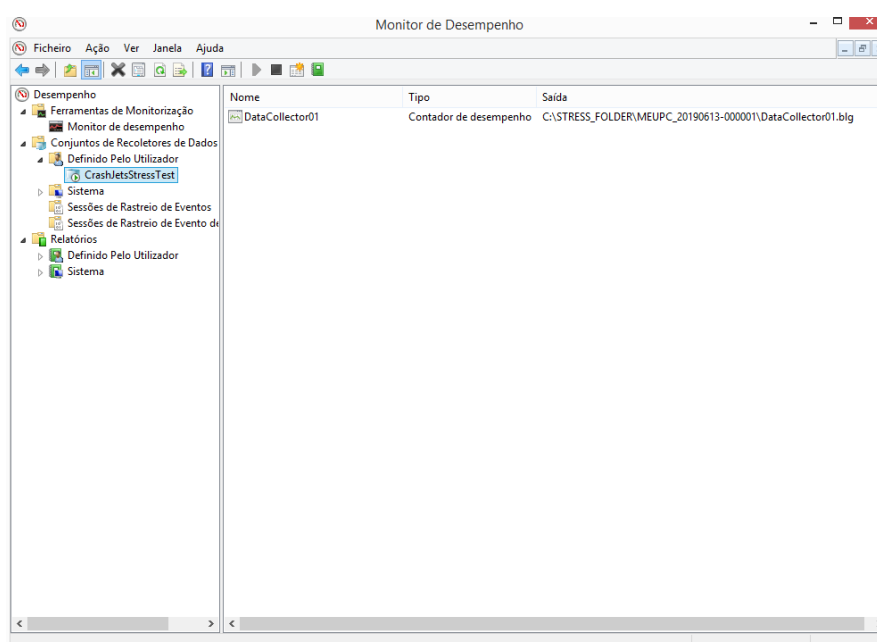


Figura 30 – Execução do *Stress Test* criado

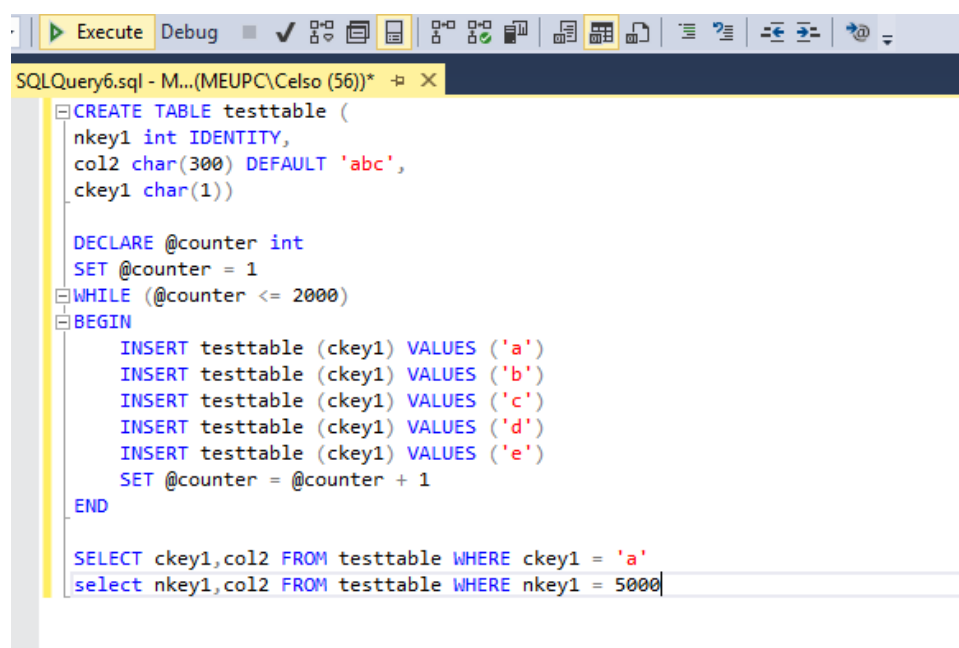


Figura 31 – Execução de *stress query* à base de dados CrashJets

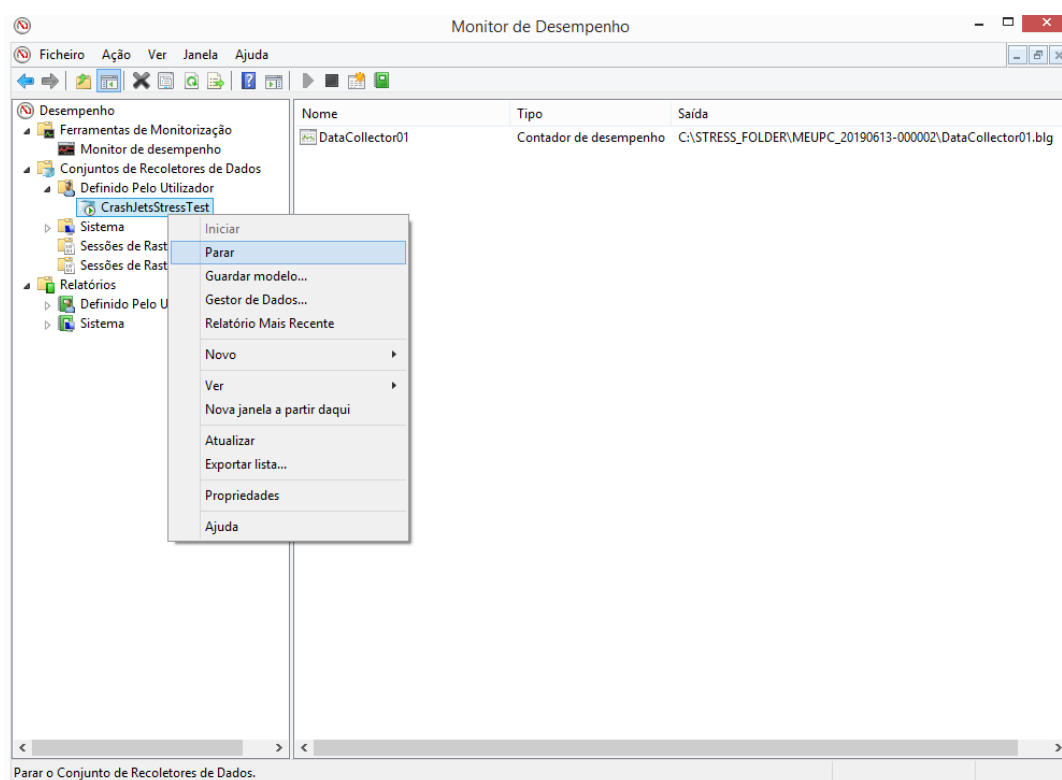


Figura 32 – Execução do *Stress Test* criado

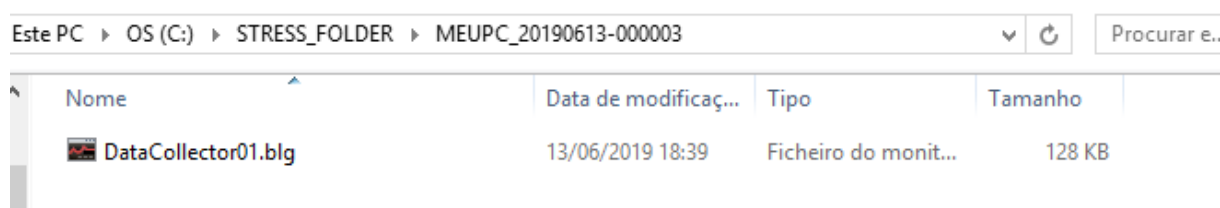


Figura 33 – Localização do *log file* resultado da execução do *Stress Test* criado

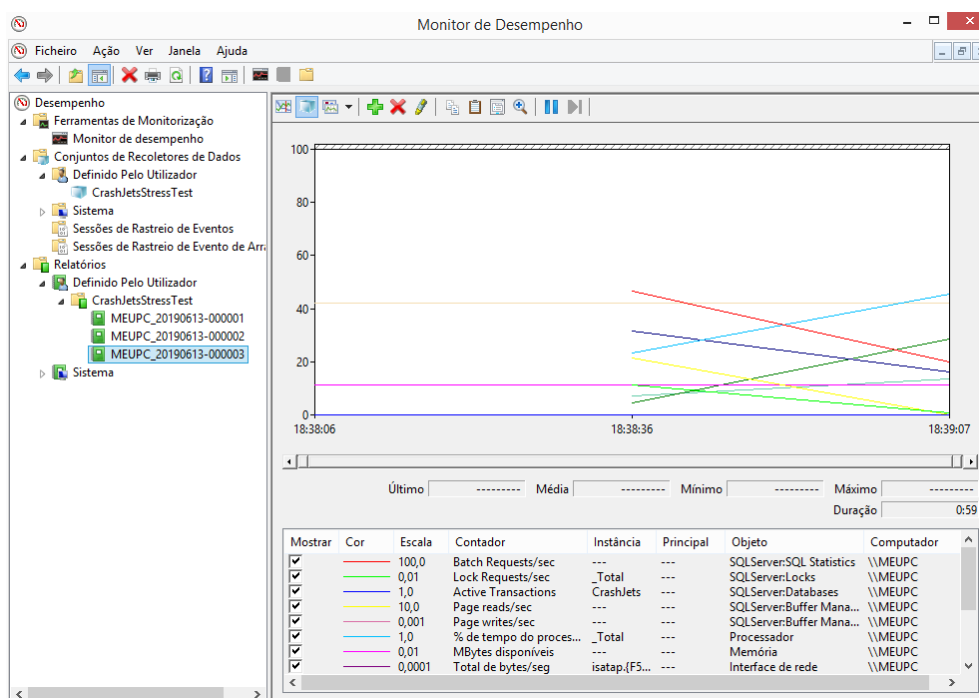


Figura 34 – Log file após execução da *stress query* à base de dados CrashJets

Demonstra-se em seguida a definição do *trace* CrashJetsTrace.trc com o *Stress Test* criado para a base de dados CrashJets:

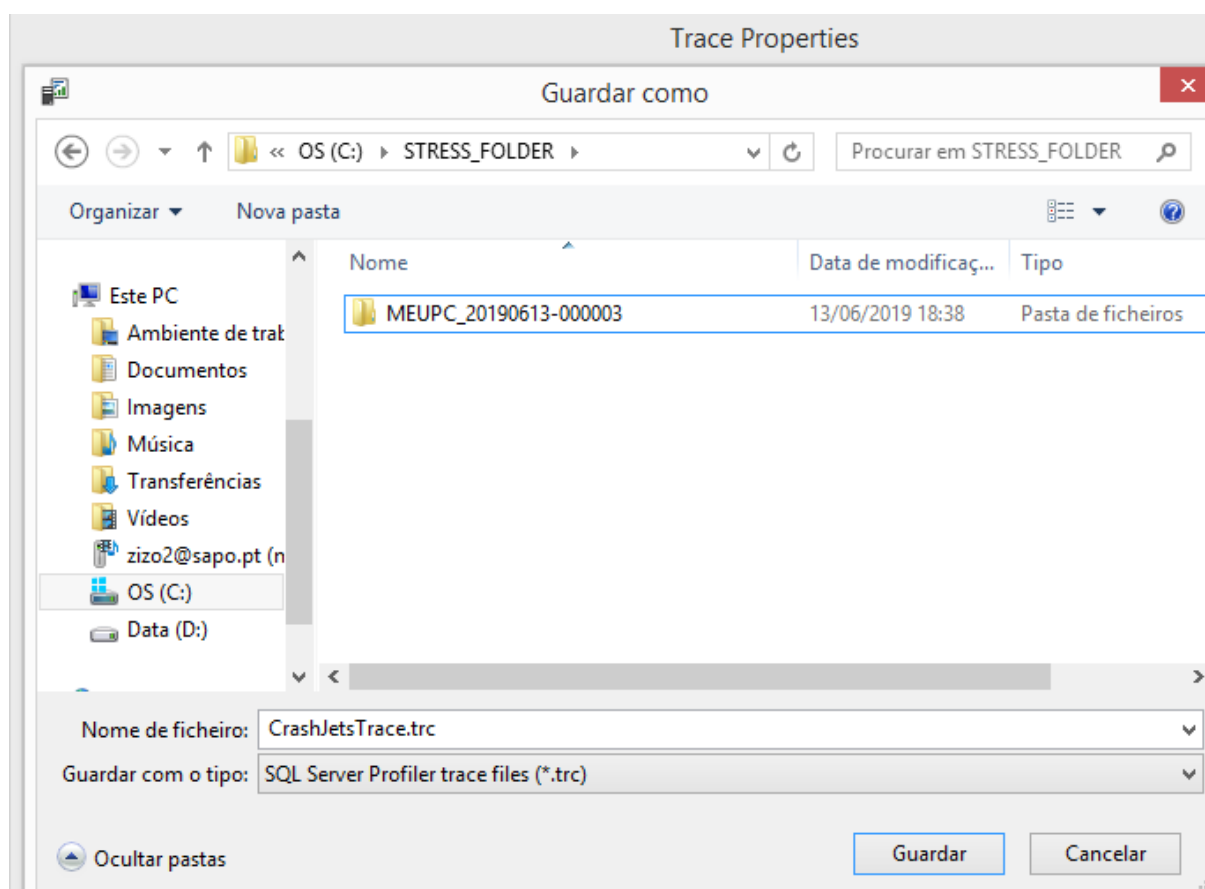


Figura 35 – Trace CrashJetsTrace.trc (localização)

Trace Properties

General | Events Selection

Trace name: CrashJetsTrace

Trace provider name: MEUPC

Trace provider type: Microsoft SQL Server "2017" version: 14.0.2014

Use the template: Standard (default)

☒ Save to file: C:\STRESS_FOLDER\CrashJetsTrace.trc

Set maximum file size (MB): 5

☒ Enable file rollover

☐ Server processes trace data

☒ Save to table: MEUPC.[CrashJets].[dbo].[CrashJets Trace]

☐ Set maximum rows (in thousands): 1

☒ Enable trace stop time: 13/06/2019 19:20:00

Run Cancelar Ajuda

Figura 36 – Propriedades do trace CrashJetsTrace

select * from dbo.CrashJetsTrace;

100 %

Results Messages

RowNumber	EventClass	TextData	ApplicationName	NTUser...	LoginName	CPU	Reads	Writes	Duration	ClientProcessID	SPID	StartTime
1	0	65528	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
2	1	65534	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	2019-06-13 19
3	2	17	-- network protocol: LPC set quoted_identifier off s...	SQLAgent - Email Log...	SQLSE...	NT SERVIC...	NULL	NULL	NULL	3344	51	2019-06-12 20
4	3	17	-- network protocol: LPC set quoted_identifier off s...	SQLAgent - Generio R...	SQLSE...	NT SERVIC...	NULL	NULL	NULL	3344	52	2019-06-12 20
5	4	17	-- network protocol: LPC set quoted_identifier on s...	SQLServerCEIP	SQLTE...	NT SERVIC...	NULL	NULL	NULL	2788	53	2019-06-13 19
6	5	17	-- network protocol: LPC set quoted_identifier off s...	SQLAgent - Alert Engi...	SQLSE...	NT SERVIC...	NULL	NULL	NULL	3344	54	2019-06-12 20
7	6	17	-- network protocol: LPC set quoted_identifier on s...	Microsoft SQL Server ...	zizo2@...	MicrosoftAc...	NULL	NULL	NULL	7260	55	2019-06-13 18
8	7	17	-- network protocol: LPC set quoted_identifier off s...	SQLAgent - Job invoc...	SQLSE...	NT SERVIC...	NULL	NULL	NULL	3344	57	2019-06-13 07
9	8	13	SELECT NTTesting Connection...	SQLAgent - Alert Engi...	SQLSE...	NT SERVIC...	NULL	NULL	NULL	3344	54	2019-06-13 19
10	9	12	SELECT NTTesting Connection...	SQLAgent - Alert Engi...	SQLSE...	NT SERVIC...	0	0	0	319	3344	2019-06-13 19
11	10	13	EXECUTE msdb.dbo.sp_sqlagent_get_perf_counters	SQLAgent - Alert Engi...	SQLSE...	NT SERVIC...	NULL	NULL	NULL	3344	54	2019-06-13 19
12	11	12	EXECUTE msdb.dbo.sp_sqlagent_get_perf_counters	SQLAgent - Alert Engi...	SQLSE...	NT SERVIC...	125	545	0	158096	3344	2019-06-13 19
13	12	13	SELECT NTTesting Connection...	SQLAgent - Alert Engi...	SQLSE...	NT SERVIC...	NULL	NULL	NULL	3344	54	2019-06-13 19
14	13	12	SELECT NTTesting Connection...	SQLAgent - Alert Engi...	SQLSE...	NT SERVIC...	0	0	0	306	3344	2019-06-13 19
15	14	10	exec sp_executesql N'SELECT o.id, o.name, n.notif...	SQLAgent - Alert Engi...	SQLSE...	NT SERVIC...	32	352	0	23321	3344	2019-06-13 19
16	15	13	SELECT NTTesting Connection...	SQLAgent - Alert Engi...	SQLSE...	NT SERVIC...	NULL	NULL	NULL	3344	54	2019-06-13 19
17	16	12	SELECT NTTesting Connection...	SQLAgent - Alert Engi...	SQLSE...	NT SERVIC...	0	0	0	291	3344	2019-06-13 19
18	17	10	exec sp_executesql N'UPDATE msdb.dbo.sysalerts...	SQLAgent - Alert Engi...	SQLSE...	NT SERVIC...	0	17	9	4756	3344	2019-06-13 19
19	18	13	SELECT NTTesting Connection...	SQLAgent - Alert Engi...	SQLSE...	NT SERVIC...	NULL	NULL	NULL	3344	54	2019-06-13 19
20	19	12	SELECT NTTesting Connection...	SQLAgent - Alert Engi...	SQLSE...	NT SERVIC...	0	0	0	318	3344	2019-06-13 19
21	20	13	EXECUTE msdb.dbo.sp_sqlagent_get_perf_counters	SQLAgent - Alert Engi...	SQLSE...	NT SERVIC...	NULL	NULL	NULL	3344	54	2019-06-13 19
22	21	12	EXECUTE msdb.dbo.sp_sqlagent_get_perf_counters	SQLAgent - Alert Engi...	SQLSE...	NT SERVIC...	32	2	0	39767	3344	2019-06-13 19
23	22	13	SELECT NTTesting Connection...	SQLAgent - Alert Engi...	SQLSE...	NT SERVIC...	NULL	NULL	NULL	3344	54	2019-06-13 19
24	23	12	SELECT NTTesting Connection...	SQLAgent - Alert Engi...	SQLSE...	NT SERVIC...	0	0	0	350	3344	2019-06-13 19

Figura 37 – Confirmação dos resultados do trace CrashJetsTrace

1.5.1. Workload e Throughput

Tendo em conta que a empresa disponibiliza dezenas de voos no mercado português, com centenas de lugares disponíveis todos os meses, espera-se uma procura média diária por reservas de até cem vezes o número de lugares diários disponíveis em voos para reserva (por exemplo, caso haja 280 lugares disponíveis em voos num determinado dia, espera-se uma procura de até 28000 clientes em simultâneo a utilizar a base de dados - operações de realização de reservas e de consulta de lotação e horário).

Estima-se que cada cliente faça até cinco operações/*queries* em simultâneo, estimando-se portanto um *throughput* na ordem das cerca de 150000 operações num curto período de tempo.

Espera-se um maior *workload* nos períodos de tempo diários entre as 12h e as 14h (UTC Lisboa) e as 17h e as 21h e um menor *workload* entre as 02h e as 07h.

1.5.2. Otimização

No que diz respeito a esta vertente, a BD foi normalizada através do método 3FN para eliminar as redundâncias dos registos, aumentar a integridade dos mesmos e o desempenho.

As *stored procedures* criadas procuram seguir as regras e convenções SQL Server. Nas mesmas procurou-se consultar e atualizar de forma atómica apenas os campos necessários à operação em causa para poupar leituras *I/O* de disco e reduzir tráfego de rede.

Os tipos de dados definidos nas várias tabelas procuram reservar o mínimo de espaço de memória para a informação armazenada nesses campos (por exemplo, *nvarchar*(100) para o campo nome do passageiro da tabela passageiro).

Definiu-se a realização de *backups* e *restores* para períodos de menor *workload* esperado.

1.5.3. Monitorização

Procura-se implementar uma política de monitorização do estado da base de dados com vista a obter os seguintes ganhos:

- Aumentar a eficácia e eficiência do trabalho do *Database Administrator*;
- Otimizar a utilização do SQL Server;
- Maximizar o uso dos recursos de *hardware* e *software* disponíveis;
- Prestação de serviços de maior qualidade;
- Minimizar os tempos de resposta.

Para obter tais ganhos implementaram-se vários fatores tais como:

- Criação de *jobs* automáticos para controlo do tamanho do *log file* e *data file* com alerta aos operadores da base de dados para atempada e adequada intervenção do DBA em casos de reduzido ou nulo espaço disponível nestes ficheiros que coloquem em estado inacessível a BD;
- Criação de *jobs* automáticos e plano de manutenção para realização dos *backups* para *backup devices* principal e espelho;
- Monitorização/*trace* de *counters* importantes (por exemplo, deteção de casos de consumo de recursos excessivo por *queries*/operações para intervenção atempada e adequada);
- Desabilitação da opção de *autogrowth* dos *filegroups* criando um ambiente de crescimento com maior espaço reservado e maior controlo do momento em que os aumentos de espaço dos *filegroups* são realizados;

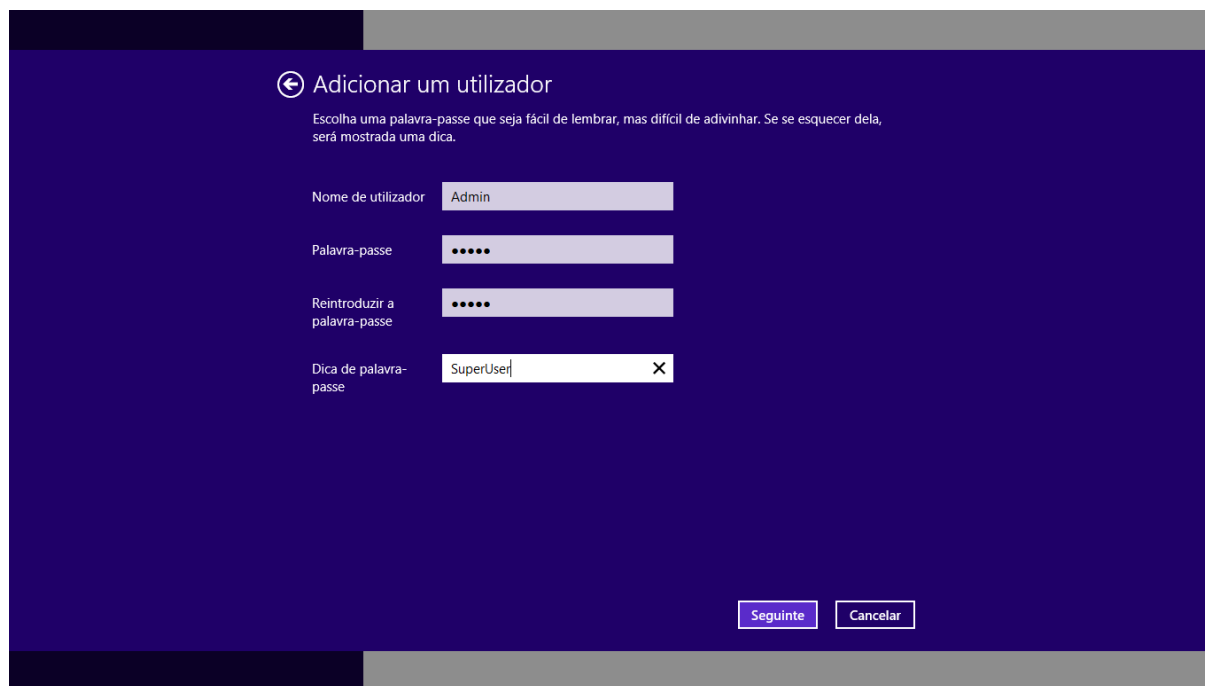
1.5.4. Recursos de Sistema

Dada a natureza e criticidade do negócio em caso de falha no seu fornecimento ao cliente, pretende-se alcançar maior *performance* global, sendo desejável ter *hardware* com uma maior capacidade física, planeando-se a utilização de vários servidores em diferentes localizações (para redundância em caso de falha no acesso a algum dos servidores), memórias RAM e processadores de última geração, discos SSD com espaço e velocidade suficientes no caso de haver necessidade de realizar várias expansões da solução de dados implementada para o negócio.

1.6. Permissões

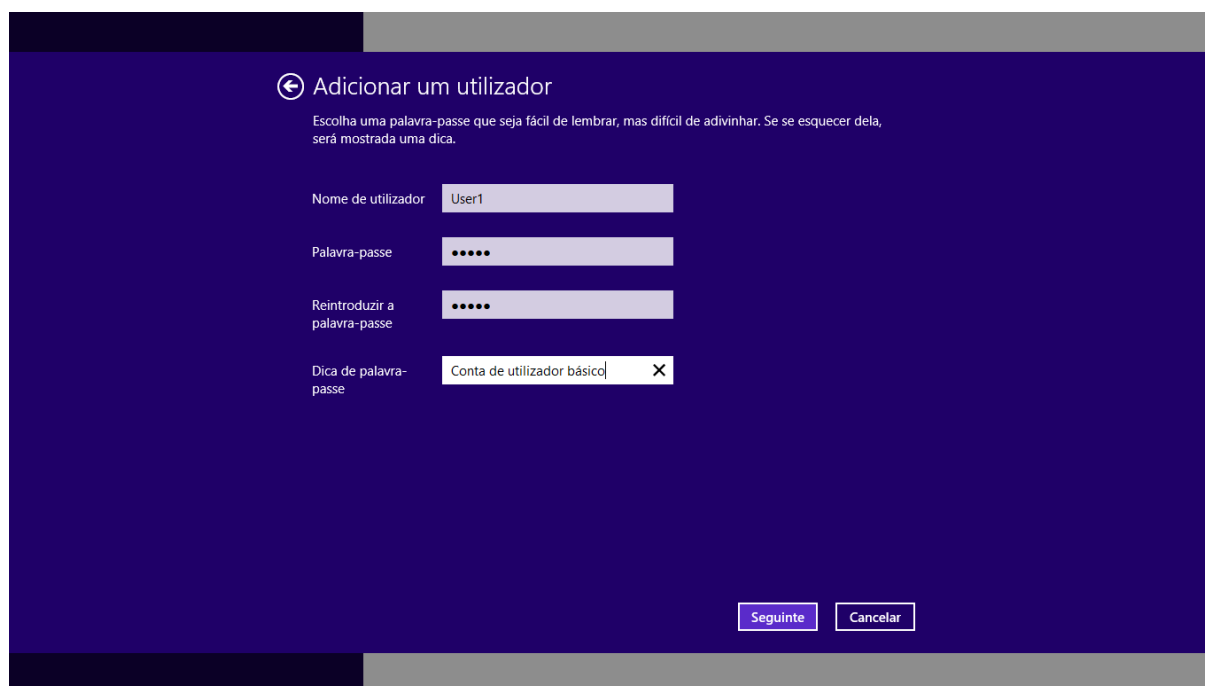
1.6.1. Users

Para realizar a gestão e controlo das permissões de acesso à base de dados foram criados três utilizadores de teste no sistema operativo Windows (modo de autenticação Windows): um utilizador de administração e dois utilizadores de negócio. Estes utilizadores permitem simular contas de utilizadores da base de dados CrashJets neste modo de autenticação aos quais foram atribuídos *roles*.



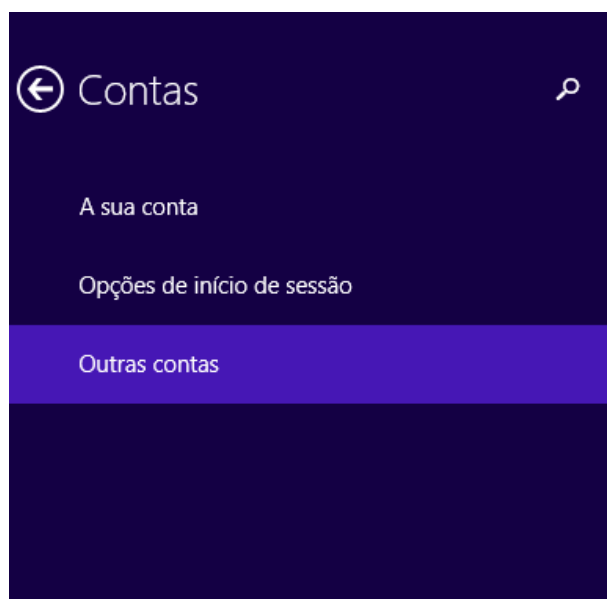
The screenshot shows the 'Adicionar um utilizador' (Add a user) window in Windows. The title bar is dark blue. The main area is white with a blue header. The text 'Adicionar um utilizador' is in blue. Below it, a message says: 'Escolha uma palavra-passe que seja fácil de lembrar, mas difícil de adivinhar. Se se esquecer dela, será mostrada uma dica.' (Choose a password that is easy to remember, but difficult to guess. If you forget it, a hint will be shown). There are four input fields: 'Nome de utilizador' (Username) with 'Admin', 'Palavra-passe' (Password) with dots, 'Reintroduzir a palavra-passe' (Re-enter password) with dots, and 'Dica de palavra-passe' (Password hint) with 'SuperUser'. At the bottom right are 'Seguinte' (Next) and 'Cancelar' (Cancel) buttons.

Figura 38 – Criação de conta de utilizador Windows de administração



The screenshot shows the 'Adicionar um utilizador' (Add a user) window in Windows. The title bar is dark blue. The main area is white with a blue header. The text 'Adicionar um utilizador' is in blue. Below it, a message says: 'Escolha uma palavra-passe que seja fácil de lembrar, mas difícil de adivinhar. Se se esquecer dela, será mostrada uma dica.' (Choose a password that is easy to remember, but difficult to guess. If you forget it, a hint will be shown). There are four input fields: 'Nome de utilizador' (Username) with 'User1', 'Palavra-passe' (Password) with dots, 'Reintroduzir a palavra-passe' (Re-enter password) with dots, and 'Dica de palavra-passe' (Password hint) with 'Conta de utilizador básico'. At the bottom right are 'Seguinte' (Next) and 'Cancelar' (Cancel) buttons.

Figura 39 – Criação de conta de utilizador Windows de negócio



Gerir outras contas

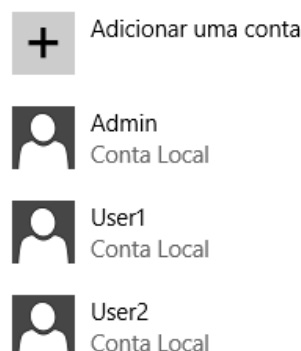


Figura 40 – Contas de utilizador Windows criadas

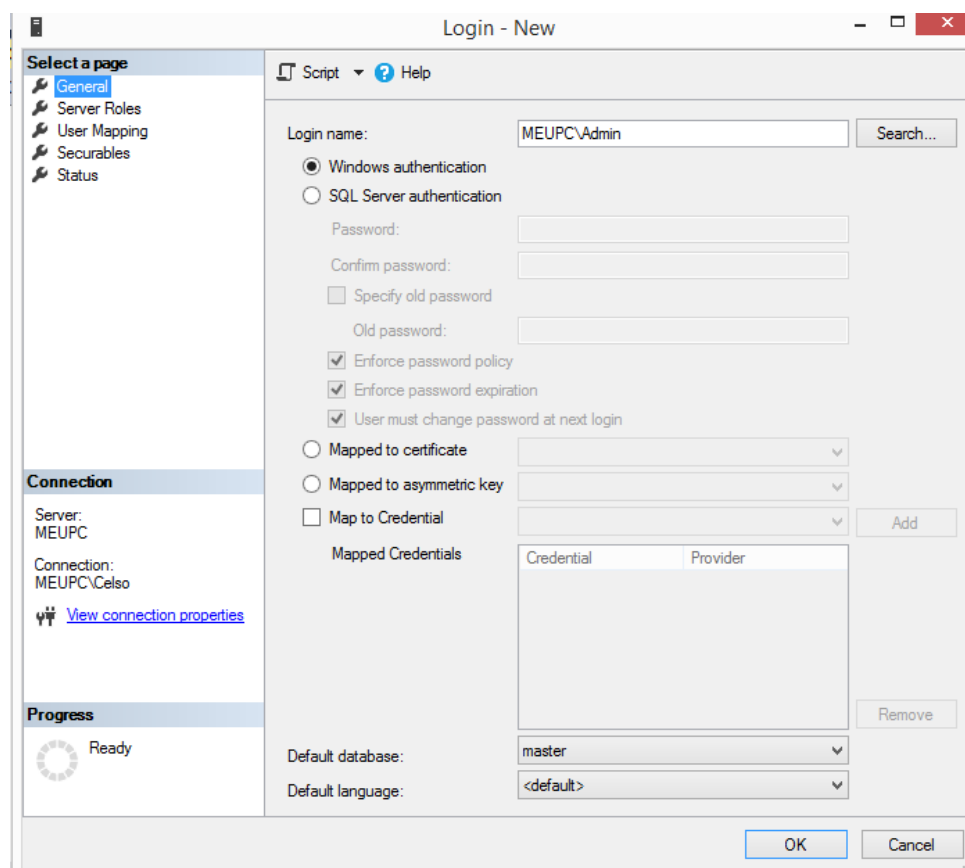


Figura 41 – Criação de novos logins no SQL Server para os utilizadores Windows criados

Para que estes utilizadores pudessem aceder ao SQL Server foram associados aos *roles* respetivos consoante as permissões necessárias.

1.6.2. Roles

Os *roles* servem para agrupar os utilizadores com o mesmo conjunto de características de permissões.

Foi atribuído o *server role sysadmin* ao utilizador de administração criado (pode executar qualquer atividade).

Foi atribuído aos dois utilizadores de negócio criados um novo *db_executor role* (podem executar *stored procedures* – para poderem executar apenas as operações relativas ao negócio como por exemplo efetuar uma reserva).

Comandos de criação do *db_executor role* na base de dados CrashJets e atribuição de permissões:

```
create role db_executor;
```

```
grant execute to db_executor;
```

SQLQuery2.sql - M...(MEUPC\Celso (56))* -p X

```
execute sp_helprole;
```

100 %

Results Messages

	RoleName	RoleId	IsAppRole
1	public	0	0
2	db_executor	5	0
3	db_owner	16384	0
4	db_accessadmin	16385	0
5	db_securityadmin	16386	0
6	db_ddladmin	16387	0
7	db_backupoperator	16389	0
8	db_datareader	16390	0
9	db_datawriter	16391	0
10	db_denydatareader	16392	0
11	db_denydatawriter	16393	0

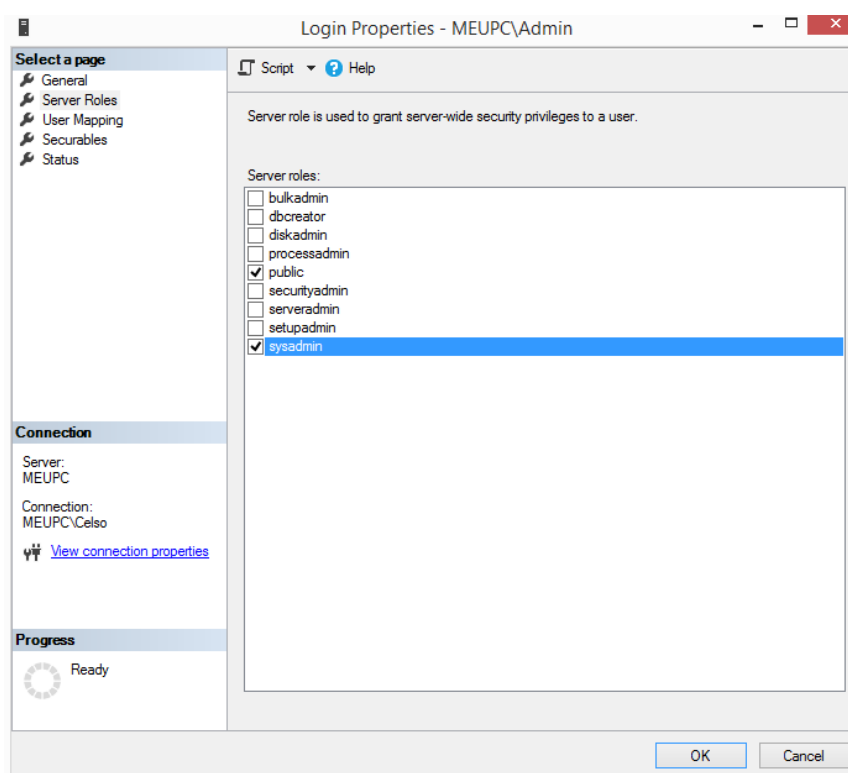


Figura 42 – Atribuição do *role* de servidor *sysadmin* ao *login* do utilizador de administração

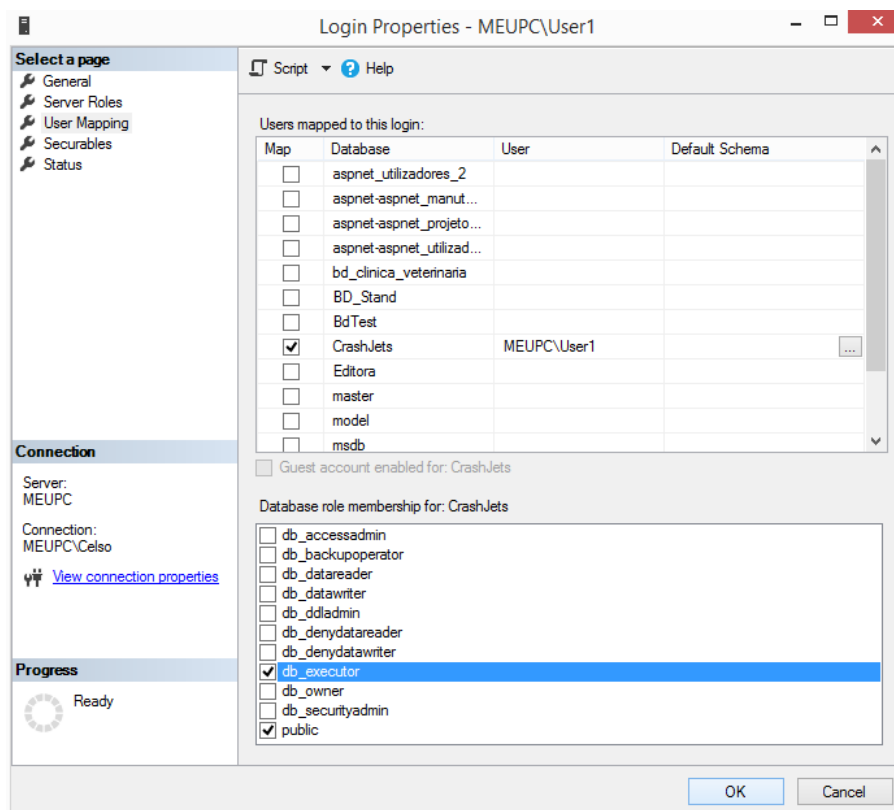


Figura 43 – Atribuição do *role* da base de dados CrashJets *db_executor* ao *login* de um dos utilizadores de negócio

2. Código T-SQL

2.1. Criação da BD

```

/*
*Use master to Create Database CrashJets
*/

USE master

GO

-----
--Create Database CrashJets--
-----

CREATE DATABASE CrashJets

ON PRIMARY (NAME = N'CrashJets_Primary', FILENAME = N'C:\CrashJets\CrashJets_Principal.mdf', SIZE = 128
MB, MAXSIZE = 256 MB, FILEGROWTH = 0),

FILEGROUP CrashJets_Filegroup DEFAULT (NAME = N'C:\CrashJets\CrashJets_Secondary01', FILENAME =
N'C:\CrashJets\MyCrashJets_Secondary01.ndf', SIZE = 128 MB, MAXSIZE = 256 MB, FILEGROWTH = 0)

```

```
LOG ON (NAME = N'CrashJets_Log', FILENAME = N'C:\CrashJets\CrashJets.ldf', SIZE = 1024 MB, MAXSIZE = 2048 MB, FILEGROWTH = 0)
```

```
GO
```

```
/*
```

```
*Use CrashJets
```

```
*/
```

```
USE CrashJets
```

```
GO
```

2.2. Criação das Tabelas

2.2.1. Aviao

```
/*
```

```
*Create Table aviao
```

```
*/
```

```
SET ANSI_NULLS ON
```

```
GO
```

```
SET QUOTED_IDENTIFIER ON
```

```
GO
```

```
CREATE TABLE [dbo].[aviao](
```

```
    [av_id] [int] IDENTITY(1,1) constraint aviao_av_id_nn NOT NULL,
```

```
    [av_lot] [int] constraint aviao_av_lot_nn NOT NULL,
```

```
    [av_nome] [nvarchar](250) constraint aviao_av_nome_nn NOT NULL,
```

```
    [data_manut] [date] constraint aviao_data_manut_nn NOT NULL,
```

```
    CONSTRAINT [aviao_av_id_pk] PRIMARY KEY CLUSTERED
```

```
(
```

```
    [av_id] ASC
```

```
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
```

```
) ON [PRIMARY]
```

```
GO
```

2.2.2. Refeicao

```
/*
```

```
*Create Table refeicao
```

```
*/
```

```
SET ANSI_NULLS ON
```

```
GO
```

```
SET QUOTED_IDENTIFIER ON
```

```
GO

CREATE TABLE [dbo].[refeicao](
    [ref_id] [int] IDENTITY(1,1) constraint refeicao_ref_id_nn NOT NULL,
    [ref_tipo] [nvarchar](100) constraint refeicao_ref_tipo_nn NOT NULL,
    CONSTRAINT [refeicao_ref_id_pk] PRIMARY KEY CLUSTERED
(
    [ref_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
```

2.2.3. Passageiro

```
/*
*Create Table passageiro
*/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[passageiro](
    [pass_id] [int] IDENTITY(1,1) constraint passageiro_pass_id_nn NOT NULL,
    [pass_nome] [nvarchar](100) constraint passageiro_pass_nome_nn NOT NULL,
    [cartao_milhas] [int] constraint passageiro_cartao_milhas_nn NOT NULL,
    CONSTRAINT [passageiro_pass_id_pk] PRIMARY KEY CLUSTERED
(
    [pass_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
```

2.2.4. Desconto

```
/*
*Create Table desconto
*/

SET ANSI_NULLS ON

GO
```



```
SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[desconto](

    [desc_id] [int] constraint desconto_desc_id_nn NOT NULL,

    [milhas_min] [bigint] constraint desconto_milhas_min_nn NOT NULL,

    [milhas_max] [bigint] constraint desconto_milhas_max_nn NOT NULL,

    [perc_desc] [int] constraint desconto_perc_desc_nn NOT NULL,

    CONSTRAINT [desconto_desc_id_pk] PRIMARY KEY CLUSTERED

(

    [desc_id] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,

ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO
```

2.2.5. Voo

```
/*

*Create Table voo

*/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[voo](

    [voo_id] [int] IDENTITY(1,1) constraint voo_voo_id_nn NOT NULL,

    [voo_dur] [time] constraint voo_voo_dur_nn NOT NULL,

    [voo_data] [datetime] constraint voo_voo_data_nn NOT NULL,

    [voo_part] [nvarchar](100) constraint voo_voo_part_nn NOT NULL,

    [voo_dest] [nvarchar](100) constraint voo_voo_dest_nn NOT NULL,

    [preco_base] [int] constraint voo_preco_base_nn NOT NULL,

    [dist_milhas] [int] constraint voo_dist_milhas_nn NOT NULL,

    [av_id] [int] constraint voo_av_id_nn NOT NULL,

    CONSTRAINT [voo_voo_id_pk] PRIMARY KEY CLUSTERED

(

    [voo_id] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,

ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO
```

2.2.6. Escala

```
/*
*Create Table escala
*/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[escala](
    [esc_id] [int] IDENTITY(1,1) constraint escala_esc_id_nn NOT NULL,
    [esc_data_ini] [datetime] constraint escala_esc_data_ini_nn NOT NULL,
    [esc_data_fim] [datetime] constraint escala_esc_data_fim_nn NOT NULL,
    [voo_id] [int] constraint escala_voo_id_nn NOT NULL,
    CONSTRAINT [escala_esc_id_pk] PRIMARY KEY CLUSTERED
(
    [esc_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

2.2.7. Funcionario

```
/*
*Create Table funcionario
*/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[funcionario](
    [func_id] [int] IDENTITY(1,1) constraint funcionario_func_id_nn NOT NULL,
    [func_nome] [nvarchar](250) constraint funcionario_func_nome_nn NOT NULL,
    [func_tipo] [nvarchar](100) constraint funcionario_func_tipo_nn NOT NULL,
    [esc_id] [int] NULL,
    CONSTRAINT [funcionario_func_id_pk] PRIMARY KEY CLUSTERED
(
    [func_id] ASC
```

```

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

```

2.2.8. Reserva

```

/*
*Create Table reserva
*/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[reserva](
    [res_id] [int] IDENTITY(1,1) constraint reserva_res_id_nn NOT NULL,
    [bilhete_emitido] [nchar](1) constraint reserva_bilhete_emitido_nn NOT NULL,
    [res_data] [datetime] constraint reserva_res_data_nn NOT NULL,
    [voo_id] [int] constraint reserva_voo_id_nn NOT NULL,
    [pass_id] [int] constraint reserva_pass_id_nn NOT NULL,
    [ref_id] [int] NULL,
    [preco_extras] [int] constraint reserva_preco_extras_nn NOT NULL default 0,
    preco_final as [dbo].[udf_GetPrecoFinal]([voo_id],[preco_extras],[pass_id]),
    CONSTRAINT [reserva_res_id_pk] PRIMARY KEY CLUSTERED
(
    [res_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

```

2.3. Adição das restrições

2.3.1. Tabela aviao

```

ALTER TABLE [dbo].[aviao] WITH CHECK ADD CONSTRAINT [aviao_maximo_avioes_ck] CHECK([av_id]<=20)

GO

ALTER TABLE [dbo].[aviao] CHECK CONSTRAINT [aviao_maximo_avioes_ck]

GO

```

```
ALTER TABLE [dbo].[aviao] WITH CHECK ADD CONSTRAINT [aviao_av_lot_ck] CHECK([av_lot] IN(80,120,140))
GO
ALTER TABLE [dbo].[aviao] CHECK CONSTRAINT [aviao_av_lot_ck]
GO
ALTER TABLE [dbo].[aviao] WITH CHECK ADD CONSTRAINT [aviao_data_manut_ck]
CHECK([data_manut]>cast(getDate() as date))
GO
ALTER TABLE [dbo].[aviao] CHECK CONSTRAINT [aviao_data_manut_ck]
GO
```

2.3.2. Tabela refeicao

```
ALTER TABLE [dbo].[refeicao] WITH CHECK ADD CONSTRAINT [refeicao_ref_tipo_ck] CHECK([ref_tipo]
IN('Normal','Dieta','Vegetariana'))
GO
ALTER TABLE [dbo].[refeicao] CHECK CONSTRAINT [refeicao_ref_tipo_ck]
GO
```

2.3.3. Tabela passageiro

```
ALTER TABLE [dbo].[passageiro] WITH CHECK ADD CONSTRAINT [desconto_cartao_milhas_ck]
CHECK([cartao_milhas]>=0)
GO
ALTER TABLE [dbo].[passageiro] CHECK CONSTRAINT [desconto_cartao_milhas_ck]
GO
```

2.3.4. Tabela desconto

```
ALTER TABLE [dbo].[desconto] WITH CHECK ADD CONSTRAINT [desconto_milhas_min_ck]
CHECK([milhas_min]>=1000)
GO
ALTER TABLE [dbo].[desconto] CHECK CONSTRAINT [desconto_milhas_min_ck]
GO
ALTER TABLE [dbo].[desconto] WITH CHECK ADD CONSTRAINT [desconto_milhas_max_ck]
CHECK([milhas_max]>[milhas_min])
GO
ALTER TABLE [dbo].[desconto] CHECK CONSTRAINT [desconto_milhas_max_ck]
GO
```

2.3.5. Tabela voo

```
ALTER TABLE [dbo].[voo] WITH CHECK ADD CONSTRAINT [voo_av_id_fk] FOREIGN KEY([av_id])
```

```
REFERENCES [dbo].[aviao] ([av_id])
GO
ALTER TABLE [dbo].[voo] CHECK CONSTRAINT [voo_av_id_fk]
GO
ALTER TABLE [dbo].[voo] WITH CHECK ADD CONSTRAINT [voo_voo_data_ck] CHECK([voo_data]>getDate())
GO
ALTER TABLE [dbo].[voo] CHECK CONSTRAINT [voo_voo_data_ck]
GO
ALTER TABLE [dbo].[voo] WITH CHECK ADD CONSTRAINT [voo_preco_base_ck] CHECK([preco_base]>0)
GO
ALTER TABLE [dbo].[voo] CHECK CONSTRAINT [voo_preco_base_ck]
GO
ALTER TABLE [dbo].[voo] WITH CHECK ADD CONSTRAINT [voo_dist_milhas_ck] CHECK([dist_milhas]>0)
GO
ALTER TABLE [dbo].[voo] CHECK CONSTRAINT [voo_dist_milhas_ck]
GO
ALTER TABLE [dbo].[voo] WITH CHECK ADD CONSTRAINT [voo_aviao_disponivel_ck]
CHECK(([dbo].[udf_AviaoEstaDisponivel]([av_id],[voo_id]))=1)
GO
ALTER TABLE [dbo].[voo] CHECK CONSTRAINT [voo_aviao_disponivel_ck]
GO
```

2.3.6. Tabela escala

```
ALTER TABLE [dbo].[escala] WITH CHECK ADD CONSTRAINT [escala_voo_id_fk] FOREIGN KEY([voo_id])
REFERENCES [dbo].[voo] ([voo_id])
GO
ALTER TABLE [dbo].[escala] CHECK CONSTRAINT [escala_voo_id_fk]
GO
ALTER TABLE [dbo].[escala] WITH CHECK ADD CONSTRAINT [escala_esc_data_fim_ck]
CHECK([esc_data_fim]>[esc_data_ini])
GO
ALTER TABLE [dbo].[escala] CHECK CONSTRAINT [escala_esc_data_fim_ck]
GO
```

2.3.7. Tabela funcionario

```
ALTER TABLE [dbo].[funcionario] WITH CHECK ADD CONSTRAINT [funcionario_maximo_funcionarios_ck]
CHECK([func_id]<=120)
GO
ALTER TABLE [dbo].[funcionario] CHECK CONSTRAINT [funcionario_maximo_funcionarios_ck]
```

```
GO

ALTER TABLE [dbo].[funcionario] WITH CHECK ADD CONSTRAINT [funcionario_esc_id_fk] FOREIGN
KEY([esc_id])

REFERENCES [dbo].[escala] ([esc_id])

GO

ALTER TABLE [dbo].[funcionario] CHECK CONSTRAINT [funcionario_esc_id_fk]

GO

ALTER TABLE [dbo].[funcionario] WITH CHECK ADD CONSTRAINT [funcionario_func_tipo_ck]
CHECK([func_tipo] IN('Tripulação','Assistente de Embarque','Apoio Administrativo'))

GO

ALTER TABLE [dbo].[funcionario] CHECK CONSTRAINT [funcionario_func_tipo_ck]

GO
```

2.3.8. Tabela reserva

```
ALTER TABLE [dbo].[reserva] WITH CHECK ADD CONSTRAINT [reserva_ref_id_fk] FOREIGN KEY([ref_id])
REFERENCES [dbo].[refeicao] ([ref_id])

GO

ALTER TABLE [dbo].[reserva] CHECK CONSTRAINT [reserva_ref_id_fk]

GO

ALTER TABLE [dbo].[reserva] WITH CHECK ADD CONSTRAINT [reserva_pass_id_fk] FOREIGN KEY([pass_id])
REFERENCES [dbo].[passageiro] ([pass_id])

GO

ALTER TABLE [dbo].[reserva] CHECK CONSTRAINT [reserva_pass_id_fk]

GO

ALTER TABLE [dbo].[reserva] WITH CHECK ADD CONSTRAINT [reserva_voo_id_fk] FOREIGN KEY([voo_id])
REFERENCES [dbo].[voo] ([voo_id])

GO

ALTER TABLE [dbo].[reserva] CHECK CONSTRAINT [reserva_voo_id_fk]

GO

ALTER TABLE [dbo].[reserva] WITH CHECK ADD CONSTRAINT [reserva_bilhete_emitido_ck]
CHECK([bilhete_emitido] IN('S','N'))

GO

ALTER TABLE [dbo].[reserva] CHECK CONSTRAINT [reserva_bilhete_emitido_ck]

GO

ALTER TABLE [dbo].[reserva] WITH CHECK ADD CONSTRAINT [reserva_preco_extras_ck]
CHECK([preco_extras]>=0)

GO

ALTER TABLE [dbo].[reserva] CHECK CONSTRAINT [reserva_preco_extras_ck]

GO
```

```

ALTER TABLE [dbo].[reserva] WITH CHECK ADD CONSTRAINT [reserva_res_data_ck]
CHECK((([dbo].[udf_IsValidResData]([res_data],[voo_id]))=1)

GO

ALTER TABLE [dbo].[reserva] CHECK CONSTRAINT [reserva_res_data_ck]

GO

ALTER TABLE [dbo].[reserva] WITH CHECK ADD CONSTRAINT [reserva_tem refeicao_ck] CHECK(
    ((([dbo].[udf_IsVooDurMaior2H]([voo_id]))!=1) AND (ref_id is null))
    OR
    ((([dbo].[udf_IsVooDurMaior2H]([voo_id]))=1) AND (ref_id IN(1,2,3))))

GO

ALTER TABLE [dbo].[reserva] CHECK CONSTRAINT [reserva_tem refeicao_ck]

GO

```

2.4. Funções desenvolvidas para a solução

2.4.1. Cálculo do preço final da reserva

```

/*
*Função para calcular o preço final da reserva segundo a fórmula:
*
*pr_fin=(pr_b+extr)*((100-perc_desc)/100)
*
*/

CREATE FUNCTION [dbo].[udf_GetPrecoFinal](@VooId int,@PrecoExtras int,@PassId int)
RETURNS INT
AS
BEGIN
    DECLARE @PrecoFinal INT;
    DECLARE @PercDesc INT;
    DECLARE @PrecoAlvo INT;
    SET @PrecoFinal=0;
    SET @PercDesc=(select (isnull(
                                                                    (select perc_desc from desconto
                                                                    where (select cartao_milhas from
passageiro where pass_id=@PassId)
                                                                    between milhas_min and milhas_max)
                                                                    ,0)));
    SET @PrecoAlvo=(select (v.preco_base+@PrecoExtras)
                                                                    from voo v where v.voo_id=@VooId);

    SET @PrecoFinal=(

```

```

        @PrecoAlvo-((@PrecoAlvo*@PercDesc)/100)
    );
    RETURN ROUND(@PrecoFinal,0);
END
GO

```

2.4.2. Verificação se a data da reserva é anterior à data do voo

```

/*
*Função para verificar se a data da reserva é anterior à data do voo
*/
CREATE FUNCTION [dbo].[udf_IsValidResData](@ResData datetime,@ResVooId int)
RETURNS BIT
AS
BEGIN
    DECLARE @Flag BIT;
    SET @Flag=0;
    IF(@ResData<(select voo_data from voo where voo_id=@ResVooId))
        BEGIN
            SET @Flag=1;
        END
    RETURN @Flag;
END
GO

```

2.4.3. Validação se a duração do voo é superior a 2 horas

```

/*
*Função para verificar se a duração do voo é superior a 2 horas
*/
CREATE FUNCTION [dbo].[udf_IsVooDurMaior2H](@ResVooId int)
RETURNS BIT
AS
BEGIN
    DECLARE @Flag BIT;
    SET @Flag=0;
    IF((select voo_dur from voo where voo_id=@ResVooId)>'02:00:00')
        BEGIN
            SET @Flag=1;
        END
    RETURN @Flag;
END
GO

```



```

        END
    RETURN @Flag;
END
GO

```

2.4.4. Validação da disponibilidade de um avião para ser utilizado num voo

```

/*
*Função para verificar se a data de partida e a data de chegada do voo não coincide com a data de
manutenção do avião.

*Um voo que se realize num determinado intervalo de datetimes só pode ser realizado por um avião que
não esteja em manutenção nesse intervalo de datetimes.

*/
CREATE FUNCTION [dbo].[udf_AviaoEstaDisponivel](@AviaoId int,@VooId int)
RETURNS BIT
AS
BEGIN
    DECLARE @Flag BIT;
    DECLARE @VooData datetime;
    DECLARE @VooDur datetime;
    DECLARE @DataFimVoo datetime;
    DECLARE @DataManut datetime;
    SET @Flag=0;
    SET @VooData=(select voo_data from voo where voo_id=@VooId);
    SET @VooDur=(select (cast(voo_dur as datetime)) from voo where voo_id=@VooId);
    SET @DataFimVoo=(@VooData+@VooDur);
    SET @DataManut=(cast((select data_manut from aviao where av_id=@AviaoId) as datetime));
    IF(
        (
            cast(@DataFimVoo as date)
            <>
            cast(@DataManut as date)
        )
        AND
        (
            cast(@VooData as date)
            <>
            cast(@DataManut as date)
        )
    )

```

```
BEGIN  
  
        SET @Flag=1;  
  
    END  
  
    RETURN @Flag;  
  
END  
  
GO
```

2.5. Procedimentos desenvolvidos para a solução

2.5.1. Listagem completa de passageiros por voo

```
/*  
*Procedimento para obter a listagem completa de passageiros por voo.  
*Inclui passageiros e funcionários da transportadora aerea.  
*/  
  
CREATE PROCEDURE [dbo].[usp_ListaPassageiros](@VooId int)  
AS  
  
    BEGIN  
  
        SELECT distinct p.[pass_id],  
            p.[pass_nome],  
            v.[voo_data] as VooData,  
            v.[voo_part] as VooPartida,  
            v.[voo_dest] as VooChegada  
        FROM [dbo].[voo] v INNER JOIN [dbo].[reserva] r  
            ON v.[voo_id]=r.[voo_id]  
            INNER JOIN [dbo].[passageiro] p  
            ON r.[pass_id]=p.[pass_id]  
  
        WHERE r.[voo_id]=@VooId  
  
        UNION ALL  
  
        SELECT distinct f.[func_id],  
            f.[func_nome],  
            v.[voo_data] as VooData,  
            v.[voo_part] as VooPartida,  
            v.[voo_dest] as VooChegada  
        FROM [dbo].[voo] v INNER JOIN [dbo].[reserva] r  
            ON v.[voo_id]=r.[voo_id]  
            INNER JOIN [dbo].[escala] e  
            ON v.[voo_id]=e.[voo_id]  
            INNER JOIN [dbo].[funcionario] f
```

```

                                ON e.[esc_id]=f.[esc_id]

                WHERE r.[voo_id]=@VooId;

        END

GO

```

2.5.2. Reserva de um voo com a validação de lotação do avião

```

/*
*Transação de reserva de um voo com validação de lotação do aviao
*/

CREATE PROCEDURE [dbo].[usp_EfetuarReserva](@VooId int,@PassId int,@RefId int,@PrecoExtras int)
AS

        BEGIN

                DECLARE @Capacidade INT;

                DECLARE @ReservasFeitas INT;

                DECLARE @CartaoMilhas INT;

                DECLARE @DistMilhas INT;

                SET @Capacidade=0;

                SET @ReservasFeitas=0;

                SET @CartaoMilhas=0;

                SET @DistMilhas=0;


                SELECT @Capacidade=a.[av_lot]

                        from aviao a inner join voo v on v.av_id=a.av_id

                        where v.[voo_id]=@VooId;


                SELECT @ReservasFeitas=(count(r.res_id))

                        from reserva r inner join voo v on r.voo_id=v.voo_id

                        where (r.voo_id=@VooId);


                IF(@ReservasFeitas<@Capacidade)

                        begin

                                BEGIN TRANSACTION;

                                IF((([dbo].[udf_IsVooDurMaior2H](@VooId))=1))

                                        begin

                                                INSERT

                                [dbo].[reserva]([bilhete_emitido],[res_data],[voo_id],[pass_id],[ref_id],[preco_extras])

                                                                VALUES

                                ('S',getdate(),@VooId,@PassId,@RefId,@PrecoExtras);

```

```

                                end
                        ELSE
                                begin
                                        INSERT
[dbo].[reserva]([bilhete_emitido],[res_data],[voo_id],[pass_id],[ref_id],[preco_extras])
                                        VALUES

                                ('S',getdate(),@VooId,@PassId,null,@PrecoExtras);
                                end
                                SELECT @CartaoMilhas=[cartao_milhas] from passageiro where
pass_id=@PassId;
                                SELECT @DistMilhas=[dist_milhas] from voo where voo_id=@VooId;
                                UPDATE passageiro
                                        set cartao_milhas=(@CartaoMilhas+@DistMilhas)
                                        where pass_id=@PassId;
                                COMMIT;
                        end
                ELSE
                        begin
                                print isnull(N'Overbooking.','(null)');
                        end
        END
GO

```

2.5.3. Obter horário e lotação de determinado voo

```

/*
*Procedimento para obter horário e lotação de determinado voo
*/
CREATE PROCEDURE [dbo].[usp_Lotacao](@VooId int)
AS
        BEGIN
                DECLARE @Capacidade INT;
                DECLARE @ReservasFeitas INT;
                SET @Capacidade=0;
                SET @ReservasFeitas=0;

                SELECT @Capacidade=a.[av_lot]
                        from aviao a inner join voo v on v.av_id=a.av_id
                        where v.[voo_id]=@VooId;
        END

```

```
SELECT @ReservasFeitas=(count(r.res_id))
        from reserva r inner join voo v on r.voo_id=v.voo_id
        where (r.voo_id=@VooId);

IF(@ReservasFeitas<@Capacidade)
    begin
        SELECT [voo_id],[voo_data],[voo_part],[voo_dest],(@Capacidade-
@ReservasFeitas) as Vagas
        from voo where voo_id=@VooId;
    end
ELSE
    begin
        SELECT [voo_id],[voo_data],[voo_part],[voo_dest], 'Lotação Esgotada' as
Vagas
        from voo where voo_id=@VooId;
    end
END
GO
```

3. Anexos

3.1. SOLUÇÃO CRASHJETS – T-SQL

```
-----
/*
*Use master to Create Database CrashJets
*/
USE master
GO

-----
--Create Database CrashJets--
-----

CREATE DATABASE CrashJets
ON PRIMARY (NAME = N'CrashJets_Primary', FILENAME = N'C:\CrashJets\CrashJets_Principal.mdf', SIZE = 128
MB, MAXSIZE = 256 MB, FILEGROWTH = 0),

FILEGROUP CrashJets_Filegroup DEFAULT (NAME = N'C:\CrashJets\CrashJets_Secondary01', FILENAME =
N'C:\CrashJets\MyCrashJets_Secondary01.ndf', SIZE = 128 MB, MAXSIZE = 256 MB, FILEGROWTH = 0)

LOG ON (NAME = N'CrashJets_Log', FILENAME = N'C:\CrashJets\CrashJets.ldf', SIZE = 1024 MB, MAXSIZE =
2048 MB, FILEGROWTH = 0)
GO
/*
*Use CrashJets
*/
USE CrashJets
GO

-----
-----
-----Funções-----
-----
/*
*Função para calcular o preço final da reserva segundo a fórmula:
*
*pr_fin=(pr_b+extr)*((100-perc_desc)/100)
*
*/
CREATE FUNCTION [dbo].[udf_GetPrecoFinal](@VooId int,@PrecoExtras int,@PassId int)
RETURNS INT
AS
    BEGIN
        DECLARE @PrecoFinal INT;
        DECLARE @PercDesc INT;
        DECLARE @PrecoAlvo INT;
        SET @PrecoFinal=0;
```

```

SET @PercDesc=(select (isnull(
                                (select perc_desc from desconto
                                where (select cartao_milhas from
passageiro where pass_id=@PassId)
                                between milhas_min and milhas_max)
                                ,0)));
SET @PrecoAlvo=(select (v.preco_base+@PrecoExtras)
                    from voo v where v.voo_id=@VooId);

SET @PrecoFinal=(
                    @PrecoAlvo-((@PrecoAlvo*@PercDesc)/100)
                );
RETURN ROUND(@PrecoFinal,0);
END

GO
/*
*Função para verificar se a data da reserva é anterior à data do voo
*/
CREATE FUNCTION [dbo].[udf_IsValidResData](@ResData datetime,@ResVooId int)
RETURNS BIT
AS
BEGIN
    DECLARE @Flag BIT;
    SET @Flag=0;
    IF(@ResData<(select voo_data from voo where voo_id=@ResVooId))
        BEGIN
            SET @Flag=1;
        END
    RETURN @Flag;
END

GO
/*
*Função para verificar se a duração do voo é superior a 2 horas
*/
CREATE FUNCTION [dbo].[udf_IsVooDurMaior2H](@ResVooId int)
RETURNS BIT
AS
BEGIN
    DECLARE @Flag BIT;
    SET @Flag=0;
    IF((select voo_dur from voo where voo_id=@ResVooId)>'02:00:00')
        BEGIN
            SET @Flag=1;
        END
    RETURN @Flag;
END

```

```
GO

/*
*Função para verificar se a data de partida e a data de chegada do voo não coincide com a data de
manutenção do avião.

*Um voo que se realize num determinado intervalo de datetimes só pode ser realizado por um avião que não
esteja em manutenção nesse intervalo de datetimes.
*/

CREATE FUNCTION [dbo].[udf_AviaoEstaDisponivel](@AviaoId int,@VooId int)
RETURNS BIT
AS
    BEGIN
        DECLARE @Flag BIT;
        DECLARE @VooData datetime;
        DECLARE @VooDur datetime;
        DECLARE @DataFimVoo datetime;
        DECLARE @DataManut datetime;
        SET @Flag=0;
        SET @VooData=(select voo_data from voo where voo_id=@VooId);
        SET @VooDur=(select (cast(voo_dur as datetime)) from voo where voo_id=@VooId);
        SET @DataFimVoo=(@VooData+@VooDur);
        SET @DataManut=(cast((select data_manut from aviao where av_id=@AviaoId) as datetime));
        IF(
            (
                cast(@DataFimVoo as date)
                <>
                cast(@DataManut as date)
            )
            AND
            (
                cast(@VooData as date)
                <>
                cast(@DataManut as date)
            )
        )
            BEGIN
                SET @Flag=1;
            END
        RETURN @Flag;
    END
GO

-----
-----
-----Procedimentos-----
-----

/*
*Procedimento para obter a listagem completa de passageiros por voo.
```



```
*Inclui passageiros e funcionários da transportadora aerea.
*/
CREATE PROCEDURE [dbo].[usp_ListaPassageiros](@VooId int)
AS
    BEGIN
        SELECT distinct p.[pass_id],
            p.[pass_nome],
            v.[voo_data] as VooData,
            v.[voo_part] as VooPartida,
            v.[voo_dest] as VooChegada
        FROM [dbo].[voo] v INNER JOIN [dbo].[reserva] r
            ON v.[voo_id]=r.[voo_id]
            INNER JOIN [dbo].[passageiro] p
            ON r.[pass_id]=p.[pass_id]
        WHERE r.[voo_id]=@VooId
        UNION ALL
        SELECT distinct f.[func_id],
            f.[func_nome],
            v.[voo_data] as VooData,
            v.[voo_part] as VooPartida,
            v.[voo_dest] as VooChegada
        FROM [dbo].[voo] v INNER JOIN [dbo].[reserva] r
            ON v.[voo_id]=r.[voo_id]
            INNER JOIN [dbo].[escala] e
            ON v.[voo_id]=e.[voo_id]
            INNER JOIN [dbo].[funcionario] f
            ON e.[esc_id]=f.[esc_id]
        WHERE r.[voo_id]=@VooId;
    END
GO
/*
*Transação de reserva de um voo com validação de lotação do aviao
*/
CREATE PROCEDURE [dbo].[usp_EfetuarReserva](@VooId int,@PassId int,@RefId int,@PrecoExtras int)
AS
    BEGIN
        DECLARE @Capacidade INT;
        DECLARE @ReservasFeitas INT;
        DECLARE @CartaoMilhas INT;
        DECLARE @DistMilhas INT;
        SET @Capacidade=0;
        SET @ReservasFeitas=0;
        SET @CartaoMilhas=0;
        SET @DistMilhas=0;
```

```

SELECT @Capacidade=a.[av_lot]
      from aviao a inner join voo v on v.av_id=a.av_id
      where v.[voo_id]=@VooId;

SELECT @ReservasFeitas=(count(r.res_id))
      from reserva r inner join voo v on r.voo_id=v.voo_id
      where (r.voo_id=@VooId);

IF(@ReservasFeitas<@Capacidade)
    begin
        BEGIN TRANSACTION;

        IF((([dbo].[udf_IsVooDurMaior2H](@VooId))=1))
            begin
                INSERT
[dbo].[reserva]([bilhete_emitido],[res_data],[voo_id],[pass_id],[ref_id],[preco_extras])
                VALUES

                ('S',getdate(),@VooId,@PassId,@RefId,@PrecoExtras);
            end
        ELSE
            begin
                INSERT
[dbo].[reserva]([bilhete_emitido],[res_data],[voo_id],[pass_id],[ref_id],[preco_extras])
                VALUES

                ('S',getdate(),@VooId,@PassId,null,@PrecoExtras);
            end
        end
        SELECT @CartaoMilhas=[cartao_milhas] from passageiro where
pass_id=@PassId;

        SELECT @DistMilhas=[dist_milhas] from voo where voo_id=@VooId;
        UPDATE passageiro
            set cartao_milhas=(@CartaoMilhas+@DistMilhas)
            where pass_id=@PassId;

        COMMIT;
    end
ELSE
    begin
        print isnull(N'Overbooking.','(null)');
    end

END

GO
/*
*Procedimento para obter horário e lotação de determinado voo
*/
CREATE PROCEDURE [dbo].[usp_Lotacao](@VooId int)
AS
    BEGIN

```

```

DECLARE @Capacidade INT;
DECLARE @ReservasFeitas INT;
SET @Capacidade=0;
SET @ReservasFeitas=0;

SELECT @Capacidade=a.[av_lot]
      from aviao a inner join voo v on v.av_id=a.av_id
      where v.[voo_id]=@VooId;

SELECT @ReservasFeitas=(count(r.res_id))
      from reserva r inner join voo v on r.voo_id=v.voo_id
      where (r.voo_id=@VooId);

IF(@ReservasFeitas<@Capacidade)
    begin
        SELECT      [voo_id],[voo_data],[voo_part],[voo_dest],(@Capacidade-
@ReservasFeitas) as Vagas
        from voo where voo_id=@VooId;
    end
ELSE
    begin
        SELECT [voo_id],[voo_data],[voo_part],[voo_dest], 'Lotação Esgotada' as
Vagas
        from voo where voo_id=@VooId;
    end
END
GO
-----
-----CREATE Tables-----
-----
/*
*Create Table aviao
*/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[aviao](
    [av_id] [int] IDENTITY(1,1) constraint aviao_av_id_nn NOT NULL,
    [av_lot] [int] constraint aviao_av_lot_nn NOT NULL,
    [av_nome] [nvarchar](250) constraint aviao_av_nome_nn NOT NULL,
    [data_manut] [date] constraint aviao_data_manut_nn NOT NULL,
    CONSTRAINT [aviao_av_id_pk] PRIMARY KEY CLUSTERED
(
    [av_id] ASC

```

```
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/*
*Create Table refeicao
*/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[refeicao](
    [ref_id] [int] IDENTITY(1,1) constraint refeicao_ref_id_nn NOT NULL,
    [ref_tipo] [nvarchar](100) constraint refeicao_ref_tipo_nn NOT NULL,
    CONSTRAINT [refeicao_ref_id_pk] PRIMARY KEY CLUSTERED
(
    [ref_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/*
*Create Table passageiro
*/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[passageiro](
    [pass_id] [int] IDENTITY(1,1) constraint passageiro_pass_id_nn NOT NULL,
    [pass_nome] [nvarchar](100) constraint passageiro_pass_nome_nn NOT NULL,
    [cartao_milhas] [int] constraint passageiro_cartao_milhas_nn NOT NULL,
    CONSTRAINT [passageiro_pass_id_pk] PRIMARY KEY CLUSTERED
(
    [pass_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/*
*Create Table desconto
*/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE TABLE [dbo].[desconto](
    [desc_id] [int] constraint desconto_desc_id_nn NOT NULL,
    [milhas_min] [bigint] constraint desconto_milhas_min_nn NOT NULL,
    [milhas_max] [bigint] constraint desconto_milhas_max_nn NOT NULL,
    [perc_desc] [int] constraint desconto_perc_desc_nn NOT NULL,
    CONSTRAINT [desconto_desc_id_pk] PRIMARY KEY CLUSTERED
(
    [desc_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/*
*Create Table voo
*/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[voo](
    [voo_id] [int] IDENTITY(1,1) constraint voo_voo_id_nn NOT NULL,
    [voo_dur] [time] constraint voo_voo_dur_nn NOT NULL,
    [voo_data] [datetime] constraint voo_voo_data_nn NOT NULL,
    [voo_part] [nvarchar](100) constraint voo_voo_part_nn NOT NULL,
    [voo_dest] [nvarchar](100) constraint voo_voo_dest_nn NOT NULL,
    [preco_base] [int] constraint voo_preco_base_nn NOT NULL,
    [dist_milhas] [int] constraint voo_dist_milhas_nn NOT NULL,
    [av_id] [int] constraint voo_av_id_nn NOT NULL,
    CONSTRAINT [voo_voo_id_pk] PRIMARY KEY CLUSTERED
(
    [voo_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/*
*Create Table escala
*/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[escala](
    [esc_id] [int] IDENTITY(1,1) constraint escala_esc_id_nn NOT NULL,
    [esc_data_ini] [datetime] constraint escala_esc_data_ini_nn NOT NULL,
    [esc_data_fim] [datetime] constraint escala_esc_data_fim_nn NOT NULL,
```

```
[voo_id] [int] constraint escala_voo_id_nn NOT NULL,
CONSTRAINT [escala_esc_id_pk] PRIMARY KEY CLUSTERED
(
    [esc_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/*
*Create Table funcionario
*/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[funcionario](
    [func_id] [int] IDENTITY(1,1) constraint funcionario_func_id_nn NOT NULL,
    [func_nome] [nvarchar](250) constraint funcionario_func_nome_nn NOT NULL,
    [func_tipo] [nvarchar](100) constraint funcionario_func_tipo_nn NOT NULL,
    [esc_id] [int] NULL,
    CONSTRAINT [funcionario_func_id_pk] PRIMARY KEY CLUSTERED
(
    [func_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/*
*Create Table reserva
*/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[reserva](
    [res_id] [int] IDENTITY(1,1) constraint reserva_res_id_nn NOT NULL,
    [bilhete_emitido] [nchar](1) constraint reserva_bilhete_emitido_nn NOT NULL,
    [res_data] [datetime] constraint reserva_res_data_nn NOT NULL,
    [voo_id] [int] constraint reserva_voo_id_nn NOT NULL,
    [pass_id] [int] constraint reserva_pass_id_nn NOT NULL,
    [ref_id] [int] NULL,
    [preco_extras] [int] constraint reserva_preco_extras_nn NOT NULL default 0,
    preco_final as [dbo].[udf_GetPrecoFinal]([voo_id],[preco_extras],[pass_id]),
    CONSTRAINT [reserva_res_id_pk] PRIMARY KEY CLUSTERED
(
    [res_id] ASC
```

```
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
-----
-----Adicionar restrições-----
-----
ALTER TABLE [dbo].[aviao] WITH CHECK ADD CONSTRAINT [aviao_maximo_avioes_ck] CHECK([av_id]<=20)
GO
ALTER TABLE [dbo].[aviao] CHECK CONSTRAINT [aviao_maximo_avioes_ck]
GO
ALTER TABLE [dbo].[aviao] WITH CHECK ADD CONSTRAINT [aviao_av_lot_ck] CHECK([av_lot] IN(80,120,140))
GO
ALTER TABLE [dbo].[aviao] CHECK CONSTRAINT [aviao_av_lot_ck]
GO
ALTER TABLE [dbo].[aviao] WITH CHECK ADD CONSTRAINT [aviao_data_manut_ck]
CHECK([data_manut]>cast(getDate() as date))
GO
ALTER TABLE [dbo].[aviao] CHECK CONSTRAINT [aviao_data_manut_ck]
GO
ALTER TABLE [dbo].[refeicao] WITH CHECK ADD CONSTRAINT [refeicao_ref_tipo_ck] CHECK([ref_tipo]
IN('Normal','Dieta','Vegetariana'))
GO
ALTER TABLE [dbo].[refeicao] CHECK CONSTRAINT [refeicao_ref_tipo_ck]
GO
ALTER TABLE [dbo].[passageiro] WITH CHECK ADD CONSTRAINT [desconto_cartao_milhas_ck]
CHECK([cartao_milhas]>=0)
GO
ALTER TABLE [dbo].[passageiro] CHECK CONSTRAINT [desconto_cartao_milhas_ck]
GO
ALTER TABLE [dbo].[desconto] WITH CHECK ADD CONSTRAINT [desconto_milhas_min_ck]
CHECK([milhas_min]>=1000)
GO
ALTER TABLE [dbo].[desconto] CHECK CONSTRAINT [desconto_milhas_min_ck]
GO
ALTER TABLE [dbo].[desconto] WITH CHECK ADD CONSTRAINT [desconto_milhas_max_ck]
CHECK([milhas_max]>[milhas_min])
GO
ALTER TABLE [dbo].[desconto] CHECK CONSTRAINT [desconto_milhas_max_ck]
GO
ALTER TABLE [dbo].[voo] WITH CHECK ADD CONSTRAINT [voo_av_id_fk] FOREIGN KEY([av_id])
REFERENCES [dbo].[aviao] ([av_id])
GO
ALTER TABLE [dbo].[voo] CHECK CONSTRAINT [voo_av_id_fk]
GO
ALTER TABLE [dbo].[voo] WITH CHECK ADD CONSTRAINT [voo_voo_data_ck] CHECK([voo_data]>getDate())
```

```
GO
ALTER TABLE [dbo].[voo] CHECK CONSTRAINT [voo_voo_data_ck]
GO
ALTER TABLE [dbo].[voo] WITH CHECK ADD CONSTRAINT [voo_preco_base_ck] CHECK([preco_base]>0)
GO
ALTER TABLE [dbo].[voo] CHECK CONSTRAINT [voo_preco_base_ck]
GO
ALTER TABLE [dbo].[voo] WITH CHECK ADD CONSTRAINT [voo_dist_milhas_ck] CHECK([dist_milhas]>0)
GO
ALTER TABLE [dbo].[voo] CHECK CONSTRAINT [voo_dist_milhas_ck]
GO
ALTER TABLE [dbo].[voo] WITH CHECK ADD CONSTRAINT [voo_aviao_disponivel_ck]
CHECK(((dbo).[udf_AviaoEstaDisponivel]([av_id],[voo_id]))=1)
GO
ALTER TABLE [dbo].[voo] CHECK CONSTRAINT [voo_aviao_disponivel_ck]
GO
ALTER TABLE [dbo].[escala] WITH CHECK ADD CONSTRAINT [escala_voo_id_fk] FOREIGN KEY([voo_id])
REFERENCES [dbo].[voo] ([voo_id])
GO
ALTER TABLE [dbo].[escala] CHECK CONSTRAINT [escala_voo_id_fk]
GO
ALTER TABLE [dbo].[escala] WITH CHECK ADD CONSTRAINT [escala_esc_data_fim_ck]
CHECK([esc_data_fim]>[esc_data_ini])
GO
ALTER TABLE [dbo].[escala] CHECK CONSTRAINT [escala_esc_data_fim_ck]
GO
ALTER TABLE [dbo].[funcionario] WITH CHECK ADD CONSTRAINT [funcionario_maximo_funcionarios_ck]
CHECK([func_id]<=120)
GO
ALTER TABLE [dbo].[funcionario] CHECK CONSTRAINT [funcionario_maximo_funcionarios_ck]
GO
ALTER TABLE [dbo].[funcionario] WITH CHECK ADD CONSTRAINT [funcionario_esc_id_fk] FOREIGN KEY([esc_id])
REFERENCES [dbo].[escala] ([esc_id])
GO
ALTER TABLE [dbo].[funcionario] CHECK CONSTRAINT [funcionario_esc_id_fk]
GO
ALTER TABLE [dbo].[funcionario] WITH CHECK ADD CONSTRAINT [funcionario_func_tipo_ck] CHECK([func_tipo]
IN('Tripulação','Assistente de Embarque','Apoio Administrativo'))
GO
ALTER TABLE [dbo].[funcionario] CHECK CONSTRAINT [funcionario_func_tipo_ck]
GO
ALTER TABLE [dbo].[reserva] WITH CHECK ADD CONSTRAINT [reserva_ref_id_fk] FOREIGN KEY([ref_id])
REFERENCES [dbo].[refeicao] ([ref_id])
GO
ALTER TABLE [dbo].[reserva] CHECK CONSTRAINT [reserva_ref_id_fk]
GO
ALTER TABLE [dbo].[reserva] WITH CHECK ADD CONSTRAINT [reserva_pass_id_fk] FOREIGN KEY([pass_id])
```



```

REFERENCES [dbo].[passageiro] ([pass_id])
GO
ALTER TABLE [dbo].[reserva] CHECK CONSTRAINT [reserva_pass_id_fk]
GO
ALTER TABLE [dbo].[reserva] WITH CHECK ADD CONSTRAINT [reserva_voo_id_fk] FOREIGN KEY([voo_id])
REFERENCES [dbo].[voo] ([voo_id])
GO
ALTER TABLE [dbo].[reserva] CHECK CONSTRAINT [reserva_voo_id_fk]
GO
ALTER TABLE [dbo].[reserva] WITH CHECK ADD CONSTRAINT [reserva_bilhete_emitido_ck]
CHECK([bilhete_emitido] IN('S','N'))
GO
ALTER TABLE [dbo].[reserva] CHECK CONSTRAINT [reserva_bilhete_emitido_ck]
GO
ALTER TABLE [dbo].[reserva] WITH CHECK ADD CONSTRAINT [reserva_preco_extras_ck]
CHECK([preco_extras]>=0)
GO
ALTER TABLE [dbo].[reserva] CHECK CONSTRAINT [reserva_preco_extras_ck]
GO
ALTER TABLE [dbo].[reserva] WITH CHECK ADD CONSTRAINT [reserva_res_data_ck]
CHECK(((dbo).[udf_IsValidResData]([res_data],[voo_id]))=1)
GO
ALTER TABLE [dbo].[reserva] CHECK CONSTRAINT [reserva_res_data_ck]
GO
ALTER TABLE [dbo].[reserva] WITH CHECK ADD CONSTRAINT [reserva_tem_refeicao_ck] CHECK(
        ((([dbo].[udf_IsVooDurMaior2H]([voo_id]))!=1) AND (ref_id is null))
        OR
        ((([dbo].[udf_IsVooDurMaior2H]([voo_id]))=1) AND (ref_id IN(1,2,3))))
GO
ALTER TABLE [dbo].[reserva] CHECK CONSTRAINT [reserva_tem_refeicao_ck]
GO

```

3.2. Inserção dos dados nas tabelas – T-SQL

3.2.1. Tabela aviao

```

/*
*Insert's tabela aviao
*/
INSERT INTO [dbo].[aviao]([av_lot],[av_nome],[data_manut]) VALUES
        (80,N'Hercules','2019-07-16'),
        (140,N'Tiger','2019-07-17'),
        (120,N'Falcon','2019-07-18'),
        (120,N'Eagle','2019-07-19'),
        (120,N'Victory','2019-07-19'),
        (80,N'Airbus','2019-07-20'),

```

```
(80,N'Hammer','2019-07-21'),
(140,N'Boeing','2019-07-22'),
(80,N'Orange','2019-07-23'),
(140,N'Pumpkin','2019-07-24'),
(120,N'Spirit','2019-07-25'),
(140,N'Apollo','2019-07-26'),
(120,N'Pilot','2019-07-26'),
(80,N'Rocket','2019-07-27'),
(80,N'Cessna','2019-07-28'),
(80,N'Raptor','2019-07-29'),
(80,N'Phantom','2019-07-30'),
(120,N'Tomcat','2019-07-30'),
(140,N'Blackbird','2019-07-16'),
(80,N'Concorde','2019-07-17')
```

GO

3.2.2. Tabela refeicao

```
/*
*Insert's tabela refeicao
*/
INSERT [dbo].[refeicao]([ref_tipo]) VALUES
    (N'Normal'),
    (N'Dieta'),
    (N'Vegetariana')
```

GO

3.2.3. Tabela passageiro

```
/*
*Insert's tabela passageiro
*/
INSERT [dbo].[passageiro]([pass_nome],[cartao_milhas]) VALUES
    (N'João Marques',250),
    (N'Vitor Gelásio',500),
    (N'Artur Ramos',5000),
    (N'Alberto Cardoso',300),
    (N'Nuno Sousa',2500),
    (N'António Costa',250),
    (N'Ana Cunha',600),
    (N'José Sócrates',60000),
    (N'Vitor Sobral',1000),
    (N'Mister T',10000),
    (N'Jonh Cooper',500),
    (N'Ruby Riott',5000),
```

(N'Liliana Jones',2000),
(N'Paulo Cunha',1000),
(N'Estela Moura',8000),
(N'Andy Gonzalez',23000),
(N'Mia Rose',8000),
(N'Pimpinha Jardim',15000),
(N'Abilio Curto',5000),
(N'Holly Molly',900),
(N'Antonieta Padilha',800),
(N'Paulo Boavida',700),
(N'Bob Esponja',12000),
(N'Brigite Bardot',5000),
(N'Bruna Castro',1000),
(N'Ciro Gomes',20000),
(N'Cheila Pereira',90000),
(N'Quintino Aires',9000),
(N'Chico Esperto',50),
(N'Mister Kaizer',800000),
(N'João Félix',1000000),
(N'Santana Lopes',500000),
(N'Damião Taveira',1000),
(N'Eliseu Antunes',3000),
(N'Horácio Moura',6000),
(N'Eduardo Valério',10000),
(N'João Sousa',550000),
(N'Pablo Aimar',2000000),
(N'Vladimir Pudim',5000000),
(N'Wally Wally',9999999),
(N'Lebron James',6666993),
(N'Bruno Lage',250000),
(N'Sérgio Conceição',156000),
(N'Donald Trump',550666),
(N'Tio Patinhas',2000000),
(N'Bruno Fernandes',2500000),
(N'Paulo Futre',99999),
(N'Joaquim das Couves',2000),
(N'Bas Dost',88888),
(N'Gary Neville',50000),
(N'Luís Filipe Vieira',3000000),
(N'Pinto da Costa',9000888),
(N'Bruno de Carvalho',1000001),
(N'Manuel Vilarinho',66666),
(N'Manuella Moura Guedes',99999),
(N'Iker Casillas',33333),
(N'Sara Carbonero',55555),

(N'Moussa Marega',1110),
(N'Dolores Aveiro',20000),
(N'Van Gog',2000),
(N'Marco Horácio',500),
(N'Salvador Daqui',660),
(N'Salvador Dali',880),
(N'Vitor Baía',990),
(N'Rúben Dias',10000),
(N'Marisa Matias',35000),
(N'Catarina Furtado',8000),
(N'António Guterres',90000),
(N'José Mourinho',600000),
(N'Claúdio Ramos',600),
(N'José Rocha',50000),
(N'André The Giant',5000),
(N'Hulk Hogan',7000),
(N'Roman Reigns',8000),
(N'Tomé Estrela',550),
(N'Miguel Ângelo',990),
(N'Lewis Hamilton',940),
(N'Alan Prost',920),
(N'Didier Drogba',1000),
(N'Didier Deschamps',1000),
(N'Giorgina Rodriguez',4444440),
(N'Norberto Santos',1000),
(N'Michael Shumacher',2000),
(N'David Couthard',4400),
(N'Pedro Lamy',5550),
(N'Tiago Monteiro',880),
(N'Ana Banana',770),
(N'Baltasar Batata',6000),
(N'Valentim Loureiro',160),
(N'Xavier Bolota',3000),
(N'Bobby Lashley',220),
(N'Jonh Cena',550),
(N'Pedro Gonçalves',990),
(N'Pedro Atum',880),
(N'Alfredo Vilela',770),
(N'Miguel Albuquerque',720),
(N'Joana Salmão',560),
(N'Elma Aveiro',1110),
(N'Eliana Marques',9990),
(N'Fernando Faria',4440),
(N'Paloma Reis',5550),
(N'Vitor Pontes',8880),

(N'Raimundo Mendonça',77990),
(N'Jorge Sampaio',9960),
(N'Alexandra Solnado',1960),
(N'Madalena Iglésias',2860),
(N'Giorgio Armani',2850),
(N'Roberto Leal',6300),
(N'Joana Espirito Santo',3600),
(N'Lua Eanes',990),
(N'Violeta Aguiar',880),
(N'Papa Xico',7650),
(N'Eduardo Mendes',6590),
(N'Luigi Ferrari',5480),
(N'Madonna',10000),
(N'Inês Guedes',20000),
(N'Helena Herédia',6330),
(N'Virgílio Faria',4440),
(N'Elton Jonh',5540),
(N'Galileu Galilei',5540),
(N'Valter Pinho',1150),
(N'Carlos Moniz',7650),
(N'Pedro Morgado',5540),
(N'Raul Monteiro',1650),
(N'Jim Carrey',1980),
(N'Mateus Oliveira',1570),
(N'Gisela Serrano',1470),
(N'Teresa Sousa',1320),
(N'Filipe Cunha',1020),
(N'Nádia Navratilova',1860),
(N'Marcelo Ribas',7980),
(N'Simão Garcia',10550),
(N'Lúcia Freire',77990),
(N'Licínio Seixas',100000),
(N'Roberta Miranda',90000),
(N'Ramiro Matias',900000),
(N'António de Oliveira Salazar',1450),
(N'Lígia Serpa',4560),
(N'Sebastião Medeiros',8000),
(N'João Marçal',9000),
(N'Filomena Ferreira',7770),
(N'Fátima Costa',7170),
(N'Margarida Fialho',8880),
(N'Teófilo Esteves',9970),
(N'Xi Jinping',8870),
(N'Sabrina Isidoro',7790),
(N'Mike Pompeo',8880),

(N'Romeu Ribeiro',2000),
(N'Martin Luther King',9990),
(N'Artur Agostinho',4660),
(N'Vânia Figueira',8770),
(N'Alexa Bliss',7770),
(N'Natália Albuquerque',70000),
(N'Conde Vladimir',800000),
(N'O Monstro das Bolachas',80000),
(N'Urbano Lage',20000),
(N'Vera Fernandes',10000),
(N'Gil Jiménez',77890),
(N'Alexandre Fagundes',3000),
(N'Valdemar Brito',300),
(N'Carlos Alberto Moniz',72000),
(N'Anna Kournikova',9820),
(N'Randy Orton',7100),
(N'Jorge Andrade',9630),
(N'Francisco Guerra',3180),
(N'Triple HHH',5490),
(N'The Rock',6320),
(N'Egídio Lopes',8790),
(N'Ana Gomes',1000),
(N'Maria João Costa',52000),
(N'Fábio Soares',5000),
(N'Albertino Vieira',9000),
(N'Filipa Vilaça',1170),
(N'Jerónimo Ochoa',770),
(N'Bruno Aleixo',7770),
(N'Nuno Markl',8880),
(N'Vince McMahon',8890),
(N'Nuno Assis',9940),
(N'António Monteiro',1000),
(N'Carmina Burana',20000),
(N'Henrique Vilar',9000),
(N'Vasco Gameiro',900010),
(N'Flávio Azevedo',1120),
(N'Nicolas Castilho',778),
(N'Armando Gama',4576),
(N'Diogo Morgado',120399),
(N'Yolanda Coelho',3443435),
(N'Wilson Borba',545515),
(N'Sofia Arruda',99885),
(N'Enrique Iglésias',945189),
(N'Tozé Marreco',91951),
(N'Alberto João Jardim',61511),

```
(N'José Alberto',98495151),  
(N'Sónia Santos',61595915),  
(N'Cristina Ferreira',91195195),  
(N'Rui Rio',951951),  
(N'Quim Barreiros',191919),  
(N'Cinha Jardim',87878),  
(N'Henrique Feist',519190),  
(N'Leonardo Davintes',9844949)
```

GO

3.2.4. Tabela desconto

```
/*  
*Insert's tabela desconto  
*/  
INSERT [dbo].[desconto]([desc_id],[milhas_min],[milhas_max],[perc_desc]) VALUES  
    (1,1000,1999,1),  
    (2,2000,2999,2),  
    (3,3000,3999,3),  
    (4,4000,4999,4),  
    (5,5000,5999,5),  
    (6,6000,6999,6),  
    (7,7000,7999,7),  
    (8,8000,8999,8),  
    (9,9000,9999,9),  
    (10,10000,10999,10),  
    (11,11000,11999,12),  
    (12,12000,12999,14),  
    (13,13000,13999,16),  
    (14,14000,14999,18),  
    (15,15000,15999,20),  
    (16,16000,16999,22),  
    (17,17000,17999,24),  
    (18,18000,18999,26),  
    (19,19000,19999,28),  
    (20,20000,22499,30),  
    (21,22500,24999,33),  
    (22,25000,27499,36),  
    (23,27500,29999,39),  
    (24,30000,49999,42),  
    (25,50000,99999,45),  
    (26,100000,199999,50),  
    (27,200000,499999,55),  
    (28,500000,999999,60),  
    (29,1000000,4999999,70),
```

```
(30,5000000,9223372036854775807,80)
```

```
GO
```

3.2.5. Tabela voo

```
/*
```

```
*Insert's tabela voo
```

```
*/
```

```
INSERT      [dbo].[voo]([voo_dur],[voo_data],[voo_part],[voo_dest],[preco_base],[dist_milhas],[av_id])
VALUES

('05:00:00','2019-08-19 13:00:00',N'Lisboa,Portugal',N'Nova Iorque,EUA',1500,3372,1),
('03:00:00','2019-08-20 13:00:00',N'Rio de Janeiro,Brasil',N'Fortaleza,Brasil',800,1363,1),
('06:30:00','2019-08-21 13:00:00',N'Pequim,China',N'Detroit,EUA',2000,5869,2),
('02:00:00','2019-08-22 13:00:00',N'Tóquio,Japão',N'Vancouver,Canadá',2500,4698,3),
('02:30:00','2019-08-23 13:00:00',N'La Paz,México',N'Caracas,Venezuela',1200,2999,4),
('04:00:00','2019-08-24 13:00:00',N'Lisboa,Portugal',N'Berlim,Alemanha',600,1438,5),
('03:00:00','2019-08-25 13:00:00',N'Lisboa,Portugal',N'Paris,França',400,903,6),
('04:30:00','2019-08-26 13:00:00',N'Lisboa,Portugal',N'São Petersburgo,Rússia',1500,2249,7),
('00:30:00','2019-08-27 13:00:00',N'Lisboa,Portugal',N'Faro,Portugal',150,172,8),
('01:00:00','2019-08-28 13:00:00',N'Lisboa,Portugal',N'Funchal,Portugal',350,605,9),
('02:00:00','2019-08-29 13:00:00',N'Londres,Reino Unido',N'São Miguel,Portugal',950,1555,10),
('03:00:00','2019-08-30 13:00:00',N'Lisboa,Portugal',N'Varsóvia,Polónia',800,1716,11),
('04:00:00','2019-09-01 10:00:00',N'Moscovo,Rússia',N'Madrid,Espanha',1250,2805,12),
('05:00:00','2019-09-01 18:00:00',N'Osaka,Japão',N'Jacarta,Indonésia',1300,3390,13),
('06:00:00','2019-09-02 10:00:00',N'Buenos Aires,Argentina',N'Barcelona,Espanha',1400,6509,14),
('07:30:00','2019-09-02 18:00:00',N'Santiago,Chile',N'Paris,França',1500,7245,15),
('05:00:00','2019-09-03 10:00:00',N'Dublin,Irlanda',N'Boston,EUA',1100,2991,16),
('04:00:00','2019-09-03 18:00:00',N'São Miguel,Portugal',N'Nova York,EUA',1250,2572,17),
('03:00:00','2019-09-04 10:00:00',N'Porto,Portugal',N'Roma,Itália',800,1091,18),
('02:00:00','2019-09-04 18:00:00',N'Los Angeles,EUA',N'Portland,EUA',750,827,19),
('02:00:00','2019-09-05 10:00:00',N'Istambul,Turquia',N'Atenas,Grécia',450,349,20),
('02:00:00','2019-09-05 18:00:00',N'Helsinquia,Finlândia',N'Estocolmo,Suécia',400,298,1),
('04:00:00','2019-09-06 10:00:00',N'Porto,Portugal',N'Praia,Cabo Verde',800,2021,2),
('03:00:00','2019-09-06 18:00:00',N'Toronto,Canadá',N'Edmonton,Canadá',750,1683,3),
('01:30:00','2019-09-07 10:00:00',N'Lisboa,Portugal',N'Nova York,EUA',1350,3372,4),
('05:30:00','2019-09-07 18:00:00',N'Sydney,Austrália',N'Auckland,Nova Zelândia',1500,1341,5),
('03:30:00','2019-09-08 10:00:00',N'Lisboa,Portugal',N'Atenas,Grécia',500,1773,6),
('04:30:00','2019-09-08 18:00:00',N'Madrid,Espanha',N'Cairo,Egito',950,2083,7),
('02:00:00','2019-09-09 10:00:00',N'Camberra,Austrália',N'Sydney,Austrália',400,177,8),
('02:00:00','2019-09-09 18:00:00',N'Calgary,Canadá',N'Toronto,Canadá',1200,1686,9),
('02:00:00','2019-09-10 10:00:00',N'Mascate,Omã',N'Abu Dhabi,EAU',550,262,10),
('02:00:00','2019-09-10 18:00:00',N'Split,Croácia',N'Salzburg,Áustria',450,339,11),
('02:00:00','2019-09-11 10:00:00',N'Nuremberga,Alemanha',N'Amesterdão,Holanda',650,337,12),
('03:00:00','2019-09-11 18:00:00',N'Manchester,Reino Unido',N'Copenhaga,Dinamarca',850,611,13),
('04:00:00','2019-09-12 10:00:00',N'Luanda,Angola',N'Cairo,Egito',1700,2943,14),
('02:00:00','2019-09-12 18:00:00',N'Lisboa,Portugal',N'Barcelona,Espanha',500,626,15),
```



```
( '02:00:00', '2019-09-13 10:00:00', N'Barcelona,Espanha', N'Porto,Portugal', 400,561,16),
( '02:00:00', '2019-09-13 18:00:00', N'São Miguel,Portugal', N'Dublin,Irlanda', 950,1414,17),
( '01:00:00', '2019-09-14 10:00:00', N'Lisboa,Portugal', N'Porto,Portugal', 150,194,18),
( '04:00:00', '2019-09-14 18:00:00', N'Yakutsk,Rússia', N'San Diego,EUA', 3500,4924,19),
( '06:00:00', '2019-09-15 10:00:00', N'Kiev,Ucrânia', N'Barcelona,Espanha', 1000,1489,20),
( '07:00:00', '2019-09-15 18:00:00', N'Keflavik,Islândia', N'Ancara,Turquia', 2000,4572,1),
( '08:00:00', '2019-09-16 10:00:00', N'Cidade do Cabo,África do Sul', N'Nova York,EUA', 5000,7814,2),
( '04:00:00', '2019-09-16 18:00:00', N'Larnaca,Chipre', N'Saransk,Rússia', 1200,1445,3),
( '02:00:00', '2019-09-17 12:00:00', N'Moscovo,Rússia', N'Baku,Azerbaijão', 2500,4191,4),
( '02:00:00', '2019-09-18 12:00:00', N'Riga,Letónia', N'Oslo,Noruega', 800,524,5),
( '02:00:00', '2019-09-19 12:00:00', N'Varsóvia,Polónia', N'Berlim,Alemanha', 400,322,6),
( '02:00:00', '2019-09-20 12:00:00', N'Lisboa,Portugal', N'Barcelona,Espanha', 500,626,7),
( '01:00:00', '2019-09-21 12:00:00', N'Porto,Portugal', N'Lisboa,Portugal', 150,194,8),
( '03:00:00', '2019-09-22 12:00:00', N'Vancouver,Canadá', N'Chicago,EUA', 1500,1773,9)
```

GO

3.2.6. Tabela escala

/*

*Insert's tabela escala

*/

```
INSERT [dbo].[escala]([esc_data_ini],[esc_data_fim],[voo_id]) VALUES
```

```
( '2019-08-19 13:00:00', '2019-08-19 18:00:00', 1),
( '2019-08-20 13:00:00', '2019-08-20 16:00:00', 2),
( '2019-08-21 13:00:00', '2019-08-21 19:30:00', 3),
( '2019-08-22 13:00:00', '2019-08-22 15:00:00', 4),
( '2019-08-23 13:00:00', '2019-08-23 15:30:00', 5),
( '2019-08-24 13:00:00', '2019-08-24 17:00:00', 6),
( '2019-08-25 13:00:00', '2019-08-25 16:00:00', 7),
( '2019-08-26 13:00:00', '2019-08-26 17:30:00', 8),
( '2019-08-27 13:00:00', '2019-08-27 13:30:00', 9),
( '2019-08-28 13:00:00', '2019-08-28 14:00:00', 10),
( '2019-08-29 13:00:00', '2019-08-29 15:00:00', 11),
( '2019-08-30 13:00:00', '2019-08-30 16:00:00', 12),
( '2019-09-01 10:00:00', '2019-09-01 14:00:00', 13),
( '2019-09-01 18:00:00', '2019-09-01 23:00:00', 14),
( '2019-09-02 10:00:00', '2019-09-02 16:00:00', 15),
( '2019-09-02 18:00:00', '2019-09-03 01:30:00', 16),
( '2019-09-03 10:00:00', '2019-09-03 15:00:00', 17),
( '2019-09-03 18:00:00', '2019-09-03 22:00:00', 18),
( '2019-09-04 10:00:00', '2019-09-04 13:00:00', 19),
( '2019-09-04 18:00:00', '2019-09-04 20:00:00', 20),
( '2019-09-05 10:00:00', '2019-09-05 12:00:00', 21),
( '2019-09-05 18:00:00', '2019-09-05 20:00:00', 22),
( '2019-09-06 10:00:00', '2019-09-06 14:00:00', 23),
```

```
( '2019-09-06 18:00:00', '2019-09-06 21:00:00', 24),
( '2019-09-07 10:00:00', '2019-09-07 11:30:00', 25),
( '2019-09-07 18:00:00', '2019-09-07 23:30:00', 26),
( '2019-09-08 10:00:00', '2019-09-08 13:30:00', 27),
( '2019-09-08 18:00:00', '2019-09-08 22:30:00', 28),
( '2019-09-09 10:00:00', '2019-09-09 12:00:00', 29),
( '2019-09-09 18:00:00', '2019-09-09 20:00:00', 30),
( '2019-09-10 10:00:00', '2019-09-10 12:00:00', 31),
( '2019-09-10 18:00:00', '2019-09-10 20:00:00', 32),
( '2019-09-11 10:00:00', '2019-09-11 12:00:00', 33),
( '2019-09-11 18:00:00', '2019-09-11 21:00:00', 34),
( '2019-09-12 10:00:00', '2019-09-12 14:00:00', 35),
( '2019-09-12 18:00:00', '2019-09-12 20:00:00', 36),
( '2019-09-13 10:00:00', '2019-09-13 12:00:00', 37),
( '2019-09-13 18:00:00', '2019-09-13 20:00:00', 38),
( '2019-09-14 10:00:00', '2019-09-14 11:00:00', 39),
( '2019-09-14 18:00:00', '2019-09-14 22:00:00', 40),
( '2019-09-15 10:00:00', '2019-09-15 16:00:00', 41),
( '2019-09-15 18:00:00', '2019-09-16 01:00:00', 42),
( '2019-09-16 10:00:00', '2019-09-16 18:00:00', 43),
( '2019-09-16 18:00:00', '2019-09-16 22:00:00', 44),
( '2019-09-17 12:00:00', '2019-09-17 14:00:00', 45),
( '2019-09-18 12:00:00', '2019-09-18 14:00:00', 46),
( '2019-09-19 12:00:00', '2019-09-19 14:00:00', 47),
( '2019-09-20 12:00:00', '2019-09-20 14:00:00', 48),
( '2019-09-21 12:00:00', '2019-09-21 13:00:00', 49),
( '2019-09-22 12:00:00', '2019-09-22 15:00:00', 50)
```

GO

3.2.7. Tabela funcionario

```
/*
*Insert's tabela funcionario
*/
INSERT [dbo].[funcionario]([func_nome],[func_tipo],[esc_id]) VALUES
(N'João Sousa',N'Tripulação',1),
(N'Laura Rebelo',N'Tripulação',1),
(N'Neymar Junior',N'Tripulação',2),
(N'Leonel Messi',N'Tripulação',2),
(N'Joaquim Simões',N'Tripulação',3),
(N'Pedro Silvestre',N'Tripulação',3),
(N'Mariana Valente',N'Tripulação',4),
(N'Mário Morgado',N'Tripulação',4),
(N'Vitor Veloso',N'Tripulação',5),
(N'Vasco Vasconcelos',N'Tripulação',5),
```

(N'Valentim Loureiro',N'Tripulação',6),
(N'José Ramos',N'Tripulação',6),
(N'João Costa',N'Tripulação',7),
(N'Ana Sousa',N'Tripulação',7),
(N'Olívia Trindade',N'Tripulação',8),
(N'Pedro Sobral',N'Tripulação',8),
(N'Nádia Navratilova',N'Tripulação',9),
(N'Mários Soares',N'Tripulação',9),
(N'Emanuel Macron',N'Tripulação',10),
(N'Vanda Sintra',N'Tripulação',10),
(N'Sónia Santos',N'Tripulação',11),
(N'Renato Duarte',N'Tripulação',11),
(N'Ricardo Pantera',N'Tripulação',12),
(N'Mateus Quinta',N'Tripulação',12),
(N'Quintino Aires',N'Tripulação',13),
(N'Cristina Ribas',N'Tripulação',13),
(N'Pedro Ramos',N'Tripulação',14),
(N'Joana Jacinto',N'Tripulação',14),
(N'Inês Paixão',N'Tripulação',15),
(N'Sílvia Ourique',N'Tripulação',15),
(N'Helena Mendonça',N'Tripulação',16),
(N'João Medina',N'Tripulação',16),
(N'Elton Rúbio',N'Tripulação',17),
(N'Elma Aveiro',N'Tripulação',17),
(N'Vera Melo',N'Tripulação',18),
(N'Diana Meira',N'Tripulação',18),
(N'José Esteves',N'Tripulação',19),
(N'Rosa Mesquita',N'Tripulação',19),
(N'David Pacheco',N'Tripulação',20),
(N'Daniel Madureira',N'Tripulação',20),
(N'Celina Jesus',N'Tripulação',21),
(N'João Lago',N'Tripulação',21),
(N'Isidoro Zarco',N'Tripulação',22),
(N'Celso Guimarães',N'Tripulação',22),
(N'Vanda Gusmão',N'Tripulação',23),
(N'Pedro Mantorras',N'Tripulação',23),
(N'Ana Marques',N'Tripulação',24),
(N'Joana Galvão',N'Tripulação',24),
(N'Baltasar Rocha',N'Tripulação',25),
(N'Amanda Freitas',N'Tripulação',25),
(N'Amélia Durão',N'Tripulação',26),
(N'Abílio Sousa',N'Tripulação',26),
(N'Paulo Domingos',N'Tripulação',27),
(N'Anabela Faro',N'Tripulação',27),
(N'Samantha Craveiro',N'Tripulação',28),

(N'Cristiana Alberto',N'Tripulação',28),
(N'Sílvia Romero',N'Tripulação',29),
(N'Jerónimo Cunha',N'Tripulação',29),
(N'Inês Palmeiro',N'Tripulação',30),
(N'João Sobral',N'Tripulação',30),
(N'Carlos Mendonça',N'Tripulação',31),
(N'Lucas Infantino',N'Tripulação',31),
(N'Rudy Gobert',N'Tripulação',32),
(N'Zeferino Lopes',N'Tripulação',32),
(N'Jack Jonhson',N'Tripulação',33),
(N'Dina Aguiar',N'Tripulação',33),
(N'Golias Abreu',N'Tripulação',34),
(N'Rui Craveiro',N'Tripulação',34),
(N'Michael Resende',N'Tripulação',35),
(N'Daniel Silva',N'Tripulação',35),
(N'Catarina Platini',N'Tripulação',36),
(N'Manuel Gião',N'Tripulação',36),
(N'Gianni Landim',N'Tripulação',37),
(N'Henrique Faro',N'Tripulação',37),
(N'Vanessa Miranda',N'Tripulação',38),
(N'Xanana Gusmão',N'Tripulação',38),
(N'António Oliveira',N'Tripulação',39),
(N'Didier Drogba',N'Tripulação',39),
(N'Éder Lopes',N'Tripulação',40),
(N'Gonçalo Guedes',N'Tripulação',40),
(N'Artur Agostinho',N'Tripulação',41),
(N'Durão Barroso',N'Tripulação',41),
(N'Manuel Queiroz',N'Tripulação',42),
(N'Domingos Andrade',N'Tripulação',42),
(N'Susana Santos',N'Tripulação',43),
(N'Soraia Dragão',N'Tripulação',43),
(N'Francisco Índia',N'Tripulação',44),
(N'Paulo Guerra',N'Tripulação',44),
(N'Jalamed Tarrafal',N'Tripulação',45),
(N'Albertino Guardiola',N'Tripulação',45),
(N'Bárbara Bandeira',N'Tripulação',46),
(N'Viktor Orban',N'Tripulação',46),
(N'Michelle Obama',N'Tripulação',47),
(N'Pedro Lamy',N'Tripulação',47),
(N'Junior Kane',N'Tripulação',48),
(N'José Fonte',N'Tripulação',48),
(N'Homero Silva',N'Tripulação',49),
(N'Ana Akmed',N'Tripulação',49),
(N'Pedro Coelho',N'Tripulação',50),
(N'Sónia Araújo',N'Tripulação',50),

```
(N'Teresa Sousa',N'Assistente de Embarque',null),
(N'Judite Resende',N'Assistente de Embarque',null),
(N'João Rebelo',N'Assistente de Embarque',null),
(N'Carlos Osório',N'Assistente de Embarque',null),
(N'Graça Meireles',N'Assistente de Embarque',null),
(N'Jorge Madeira',N'Assistente de Embarque',null),
(N'César Augusto',N'Assistente de Embarque',null),
(N'Benjamim Guedes',N'Assistente de Embarque',null),
(N'Gil Gomes',N'Assistente de Embarque',null),
(N'Catarina Freixo',N'Assistente de Embarque',null),
(N'Dinis Liberato',N'Assistente de Embarque',null),
(N'Cristiano Ronaldo',N'Apoio Administrativo',null),
(N'Beatriz Rocha',N'Apoio Administrativo',null),
(N'António Salgueiro',N'Apoio Administrativo',null),
(N'Marta Torres',N'Apoio Administrativo',null),
(N'Maria Valério',N'Apoio Administrativo',null),
(N'Miguel Viana',N'Apoio Administrativo',null),
(N'Carla Rocha',N'Apoio Administrativo',null),
(N'Bárbara Andrade',N'Apoio Administrativo',null),
(N'Juvenal Malheiro',N'Apoio Administrativo',null)
```

GO

3.2.8. Tabela reserva

```
/*
*Insert's tabela reserva
*/
INSERT [dbo].[reserva]([bilhete_emitido],[res_data],[voo_id],[pass_id],[ref_id],[preco_extras]) VALUES
(N'S','2019-05-01 11:00:00',1,1,1,0),
(N'S','2019-05-01 11:00:00',1,2,2,0),
(N'S','2019-05-01 11:00:00',1,3,3,0),
(N'S','2019-05-01 11:00:00',1,4,1,0),
(N'S','2019-05-01 11:00:00',1,5,2,0),
(N'S','2019-05-01 11:00:00',1,6,3,100),
(N'S','2019-05-01 11:00:00',1,7,1,0),
(N'S','2019-05-01 11:00:00',1,8,2,0),
(N'S','2019-05-01 11:00:00',1,9,3,0),
(N'S','2019-05-01 11:00:00',1,10,1,0),
(N'S','2019-05-01 11:00:00',1,11,2,0),
(N'S','2019-05-01 11:00:00',1,12,3,0),
(N'S','2019-05-01 11:00:00',1,13,1,0),
(N'S','2019-05-01 11:00:00',1,14,2,0),
(N'S','2019-05-01 11:00:00',1,15,3,0),
(N'S','2019-05-01 11:00:00',1,16,1,0),
(N'S','2019-05-01 11:00:00',1,17,2,0),
```

(N'S', '2019-05-01 11:00:00', 1, 18, 2, 80),
(N'S', '2019-05-01 11:00:00', 1, 19, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 20, 2, 0),
(N'S', '2019-05-01 11:00:00', 1, 21, 3, 0),
(N'S', '2019-05-01 11:00:00', 1, 22, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 23, 2, 0),
(N'S', '2019-05-01 11:00:00', 1, 24, 3, 200),
(N'S', '2019-05-01 11:00:00', 1, 25, 1, 150),
(N'S', '2019-05-01 11:00:00', 1, 26, 2, 0),
(N'S', '2019-05-01 11:00:00', 1, 27, 3, 0),
(N'S', '2019-05-01 11:00:00', 1, 28, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 29, 2, 0),
(N'S', '2019-05-01 11:00:00', 1, 30, 3, 0),
(N'S', '2019-05-01 11:00:00', 1, 31, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 32, 2, 0),
(N'S', '2019-05-01 11:00:00', 1, 33, 3, 0),
(N'S', '2019-05-01 11:00:00', 1, 34, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 35, 2, 0),
(N'S', '2019-05-01 11:00:00', 1, 36, 3, 0),
(N'S', '2019-05-01 11:00:00', 1, 37, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 38, 2, 0),
(N'S', '2019-05-01 11:00:00', 1, 39, 3, 0),
(N'S', '2019-05-01 11:00:00', 1, 40, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 41, 2, 0),
(N'S', '2019-05-01 11:00:00', 1, 42, 3, 0),
(N'S', '2019-05-01 11:00:00', 1, 43, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 44, 2, 0),
(N'S', '2019-05-01 11:00:00', 1, 45, 3, 0),
(N'S', '2019-05-01 11:00:00', 1, 46, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 47, 2, 0),
(N'S', '2019-05-01 11:00:00', 1, 48, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 49, 3, 0),
(N'S', '2019-05-01 11:00:00', 1, 50, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 51, 2, 0),
(N'S', '2019-05-01 11:00:00', 1, 52, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 53, 3, 0),
(N'S', '2019-05-01 11:00:00', 1, 54, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 55, 2, 0),
(N'S', '2019-05-01 11:00:00', 1, 56, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 57, 3, 0),
(N'S', '2019-05-01 11:00:00', 1, 58, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 59, 2, 0),
(N'S', '2019-05-01 11:00:00', 1, 60, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 61, 3, 0),
(N'S', '2019-05-01 11:00:00', 1, 62, 1, 0),

```
(N'S', '2019-05-01 11:00:00', 1, 63, 2, 0),
(N'S', '2019-05-01 11:00:00', 1, 64, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 65, 3, 0),
(N'S', '2019-05-01 11:00:00', 1, 66, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 67, 2, 0),
(N'S', '2019-05-01 11:00:00', 1, 68, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 69, 3, 0),
(N'S', '2019-05-01 11:00:00', 1, 70, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 71, 2, 0),
(N'S', '2019-05-01 11:00:00', 1, 72, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 73, 3, 0),
(N'S', '2019-05-01 11:00:00', 1, 74, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 75, 2, 0),
(N'S', '2019-05-01 11:00:00', 1, 76, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 77, 3, 0),
(N'S', '2019-05-01 11:00:00', 1, 78, 1, 0),
(N'S', '2019-05-01 11:00:00', 1, 79, 2, 0),
(N'S', '2019-05-01 11:00:00', 1, 80, 1, 0),
(N'S', '2019-05-01 11:00:00', 2, 81, 3, 0),
(N'S', '2019-05-01 11:00:00', 3, 82, 1, 0),
(N'S', '2019-05-01 11:00:00', 4, 83, null, 0),
(N'S', '2019-05-01 11:00:00', 5, 84, 1, 0),
(N'S', '2019-05-01 11:00:00', 5, 85, 3, 0),
(N'S', '2019-05-01 11:00:00', 6, 86, 1, 0),
(N'S', '2019-05-01 11:00:00', 6, 87, 2, 0),
(N'S', '2019-05-01 11:00:00', 6, 88, 1, 0),
(N'S', '2019-05-01 11:00:00', 6, 89, 3, 0),
(N'S', '2019-05-01 11:00:00', 6, 90, 1, 0),
(N'S', '2019-05-01 11:00:00', 6, 91, 2, 0),
(N'S', '2019-05-01 11:00:00', 6, 92, 1, 0),
(N'S', '2019-05-01 11:00:00', 7, 93, 3, 0),
(N'S', '2019-05-01 11:00:00', 7, 94, 1, 0),
(N'S', '2019-05-01 11:00:00', 7, 95, 2, 0),
(N'S', '2019-05-01 11:00:00', 7, 96, 1, 250),
(N'S', '2019-05-01 11:00:00', 7, 97, 3, 0),
(N'S', '2019-05-01 11:00:00', 7, 98, 1, 0),
(N'S', '2019-05-01 11:00:00', 7, 99, 2, 0),
(N'S', '2019-05-01 11:00:00', 7, 100, 1, 0),
(N'S', '2019-05-01 11:00:00', 8, 101, 3, 0),
(N'S', '2019-05-01 11:00:00', 8, 102, 1, 0),
(N'S', '2019-05-01 11:00:00', 8, 103, 2, 0),
(N'S', '2019-05-01 11:00:00', 8, 104, 1, 0),
(N'S', '2019-05-01 11:00:00', 8, 105, 3, 0),
(N'S', '2019-05-01 11:00:00', 8, 106, 1, 0),
(N'S', '2019-05-01 11:00:00', 9, 107, null, 0),
```

```
(N'S', '2019-05-01 11:00:00', 9, 108, null, 0),
(N'S', '2019-05-01 11:00:00', 9, 109, null, 0),
(N'S', '2019-05-01 11:00:00', 10, 110, null, 0),
(N'S', '2019-05-01 11:00:00', 10, 111, null, 0),
(N'S', '2019-05-01 11:00:00', 10, 112, null, 0),
(N'S', '2019-05-01 11:00:00', 10, 113, null, 0),
(N'S', '2019-05-01 11:00:00', 11, 114, null, 0),
(N'S', '2019-05-01 11:00:00', 11, 115, null, 0),
(N'S', '2019-05-01 11:00:00', 11, 116, null, 0),
(N'S', '2019-05-01 11:00:00', 11, 117, null, 0),
(N'S', '2019-05-01 11:00:00', 12, 118, 2, 0),
(N'S', '2019-05-01 11:00:00', 12, 119, 1, 0),
(N'S', '2019-05-01 11:00:00', 12, 120, 2, 0),
(N'S', '2019-05-01 11:00:00', 12, 121, 1, 0),
(N'S', '2019-05-01 11:00:00', 13, 122, 1, 0),
(N'S', '2019-05-01 11:00:00', 13, 123, 1, 0),
(N'S', '2019-05-01 11:00:00', 13, 124, 2, 0),
(N'S', '2019-05-01 11:00:00', 13, 125, 1, 0),
(N'S', '2019-05-01 11:00:00', 14, 126, 1, 0),
(N'S', '2019-05-01 11:00:00', 14, 127, 3, 0),
(N'S', '2019-05-01 11:00:00', 14, 128, 1, 0),
(N'S', '2019-05-01 11:00:00', 14, 129, 2, 20),
(N'S', '2019-05-01 11:00:00', 14, 130, 1, 100),
(N'S', '2019-05-01 11:00:00', 14, 131, 1, 50),
(N'S', '2019-05-01 11:00:00', 15, 132, 3, 60),
(N'S', '2019-05-01 11:00:00', 15, 133, 1, 400),
(N'S', '2019-05-01 11:00:00', 15, 134, 2, 90),
(N'S', '2019-05-01 11:00:00', 15, 135, 1, 0),
(N'S', '2019-05-01 11:00:00', 16, 136, 3, 0),
(N'S', '2019-05-01 11:00:00', 16, 137, 1, 0),
(N'S', '2019-05-01 11:00:00', 17, 138, 1, 0),
(N'S', '2019-05-01 11:00:00', 17, 139, 1, 0),
(N'S', '2019-05-01 11:00:00', 18, 140, 2, 0),
(N'S', '2019-05-01 11:00:00', 18, 141, 1, 0),
(N'S', '2019-05-01 11:00:00', 18, 142, 3, 0),
(N'S', '2019-05-01 11:00:00', 19, 143, 1, 0),
(N'S', '2019-05-01 11:00:00', 20, 144, null, 0),
(N'S', '2019-05-01 11:00:00', 22, 145, null, 0),
(N'S', '2019-05-01 11:00:00', 23, 146, 1, 0),
(N'S', '2019-05-01 11:00:00', 24, 147, 1, 0),
(N'S', '2019-05-01 11:00:00', 25, 148, null, 0),
(N'S', '2019-05-01 11:00:00', 26, 149, 1, 0),
(N'S', '2019-05-01 11:00:00', 27, 150, 1, 0),
(N'S', '2019-05-01 11:00:00', 28, 151, 2, 0),
(N'S', '2019-05-01 11:00:00', 29, 152, null, 0),
```


(N'S', '2019-05-01 11:00:00', 30, 153, null, 0),
(N'S', '2019-05-01 11:00:00', 31, 154, null, 0),
(N'S', '2019-05-01 11:00:00', 32, 155, null, 0),
(N'S', '2019-05-01 11:00:00', 33, 156, null, 0),
(N'S', '2019-05-01 11:00:00', 34, 157, 1, 0),
(N'S', '2019-05-01 11:00:00', 34, 158, 3, 0),
(N'S', '2019-05-01 11:00:00', 34, 159, 2, 0),
(N'S', '2019-05-01 11:00:00', 35, 160, 3, 0),
(N'S', '2019-05-01 11:00:00', 35, 161, 1, 0),
(N'S', '2019-05-01 11:00:00', 36, 162, null, 0),
(N'S', '2019-05-01 11:00:00', 36, 163, null, 0),
(N'S', '2019-05-01 11:00:00', 37, 164, null, 0),
(N'S', '2019-05-01 11:00:00', 38, 165, null, 0),
(N'S', '2019-05-01 11:00:00', 39, 166, null, 0),
(N'S', '2019-05-01 11:00:00', 40, 167, 1, 0),
(N'S', '2019-05-01 11:00:00', 41, 168, 1, 0),
(N'S', '2019-05-01 11:00:00', 41, 169, 1, 0),
(N'S', '2019-05-01 11:00:00', 41, 170, 1, 0),
(N'S', '2019-05-01 11:00:00', 41, 171, 1, 0),
(N'S', '2019-05-01 11:00:00', 41, 172, 1, 0),
(N'S', '2019-05-01 11:00:00', 41, 173, 1, 40),
(N'S', '2019-05-01 11:00:00', 41, 174, 1, 0),
(N'S', '2019-05-01 11:00:00', 41, 175, 1, 0),
(N'S', '2019-05-01 11:00:00', 41, 176, 1, 0),
(N'S', '2019-05-01 11:00:00', 41, 177, 1, 0),
(N'S', '2019-05-01 11:00:00', 41, 178, 1, 0),
(N'S', '2019-05-01 11:00:00', 42, 179, 1, 0),
(N'S', '2019-05-01 11:00:00', 43, 180, 1, 0),
(N'S', '2019-05-01 11:00:00', 44, 181, 1, 0),
(N'S', '2019-05-01 11:00:00', 45, 182, null, 0),
(N'S', '2019-05-01 11:00:00', 46, 183, null, 0),
(N'S', '2019-05-01 11:00:00', 47, 184, null, 0),
(N'S', '2019-05-01 11:00:00', 47, 185, null, 0),
(N'S', '2019-05-01 11:00:00', 47, 186, null, 0),
(N'S', '2019-05-01 11:00:00', 47, 187, null, 0),
(N'S', '2019-05-01 11:00:00', 48, 188, null, 0),
(N'S', '2019-05-01 11:00:00', 48, 189, null, 0),
(N'S', '2019-05-01 11:00:00', 48, 190, null, 0),
(N'S', '2019-05-01 11:00:00', 49, 191, null, 0),
(N'S', '2019-05-01 11:00:00', 49, 192, null, 0),
(N'S', '2019-05-01 11:00:00', 49, 193, null, 0),
(N'S', '2019-05-01 11:00:00', 49, 194, null, 0),
(N'S', '2019-05-01 11:00:00', 49, 195, null, 0),
(N'S', '2019-05-01 11:00:00', 49, 196, null, 0),
(N'S', '2019-05-01 11:00:00', 49, 197, null, 0),

```
(N'S', '2019-05-01 11:00:00', 49, 198, null, 0),  
(N'S', '2019-05-01 11:00:00', 49, 199, null, 25),  
(N'S', '2019-05-01 11:00:00', 49, 200, null, 180)
```

GO

3.3. Ficheiros entregues

- ***CrashJets.ldf*** (Log File);
- ***CrashJets_Principal.mdf*** (Primary Data File);
- ***MyCrashJets_Secondary01.ndf*** (Secondary Data File);
- ***CrashJets.bak*** (Database Backup File);
- ***CrashJets_CreatesDBScript.sql*** (Estrutura da base de dados e seus objetos);
- ***scriptCrashJets_Q1.sql*** (Listagem completa de passageiros por voo);
- ***scriptCrashJets_Q2.sql*** (Transação de reserva de um voo com validação de lotação do avião);
- ***scriptCrashJets_Q3.sql*** (Consulta para análise de horário e lotação de determinado voo);
- ***CrashJestTrace.trc*** (Ficheiro de *trace* de *stress test* para a base de dados CrashJets).