Universidade de Trás-os-Montes e Alto Douro

Relatório Detalhado da Execução do Trabalho Prático Nº2

Licenciatura em Engenharia Informática

Técnicas Avançadas de Bases de Dados

Paulo Martins

António Marques

Realizado pelos alunos:

- Hugo Anes 68571
- Vítor Neto 68717
- João Leal 68719
- Leandro Coelho 68541

Índice

Introdução

Neste trabalho experimental 2 vamos dar continuidade ao que foi feito no trabalho experimental 1 e que tem como objetivo criar um modelo de distribuição da base de dados as políticas de segurança e acesso aos dados distribuídos, a resolução para os problemas de concorrência e a análise de otimização de questões distribuídas. Melhoramos também alguns aspetos no trabalho experimental 1 para que a base de dados tivesse um melhor funcionamento.

Enquadramento Teórico

Uma base de dados distribuída é um sistema de base de dados cujos dados se encontram fisicamente dispersos por várias máquinas, ligadas por meios de comunicação, mas integrados logicamente em que esta possui várias características tais como disponibilidade permanente, Fragmentação transparente, Duplicação transparente, Processamento de questões distribuído e Independência do hardware, sistema operativo e tecnologia de rede.

Na implementação deste trabalho iremos usar bases de dados distribuídas homogéneas, isto é, todos os nodos usam o mesmo sistema de gestão de base de dados (SGBD), fazendo lembrar um único sistema de bases de dados, mas em vez de todos os dados estarem armazenados num único repositório, os dados estão armazenados por vários repositórios ligados por meios de comunicação. Iremos usar uma abordagem "Top-down", que corresponde à divisão de uma base de dados preexistente ou, mais genericamente, de um esquema de base de dados predefinido em várias partes a armazenar em diferentes nodos.

Distribuição da base de dados

O processamento distribuído de bases de dados proporciona melhor desempenho e maior facilidade de expansão, e aumenta a disponibilidade do sistema. No entanto, devemos ter em conta que, sob determinadas condições, o desempenho e a disponibilidade do sistema também podem diminuir. Por outro lado, tem maior complexidade, tornando-se mais difícil de controlar.

Para ser possível a criação de uma base de dados distribuída iremos utilizar dois store procedure que nos irão ajudar.

sp_addlinkedserver- Este store procedure vai possibilitar a criação de um *linked server* que nos vai possibilitar utilizar dados das várias bases de dados, no nosso caso das duas bases que iremos utilizar (no servidor 1 e servidor 2).

Definimos as configurações das ligações como o tipo de serviço, e a localização do servidor.

sp_addlinkedsrvlogin – Neste store procedure definimos o login do *linked server*. No caso definimos o login "sa" e a respective password.

Por último é necessário a criação de views, são estas que vão auxiliar a utilização dos vários dados dos diferentes servidores, estas são também denominadas por tabelas virtuais pois fazem parecer que os diferentes dados aparentem ser todos da mesma tabela, dando uma experiência ao utilizador de aceder a uma única base de dados.

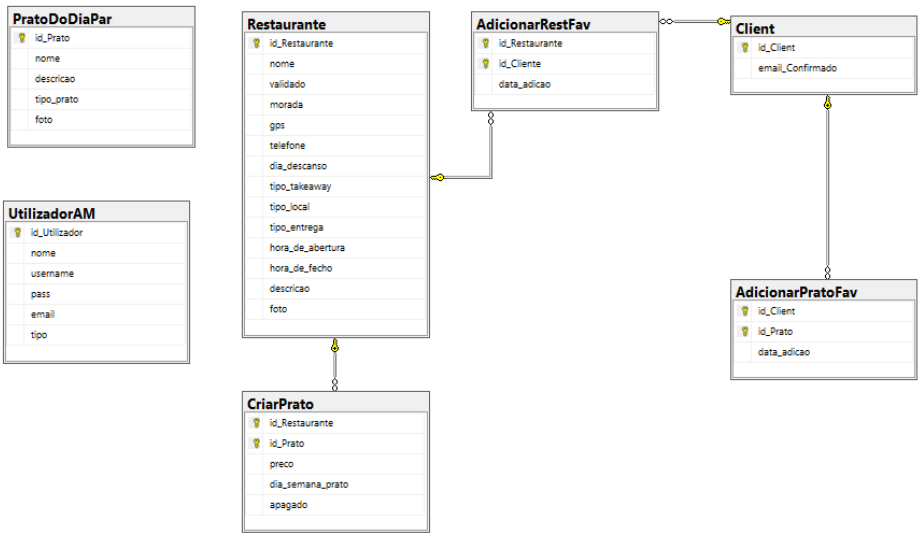

Figura 1: Diagrama servidor 1

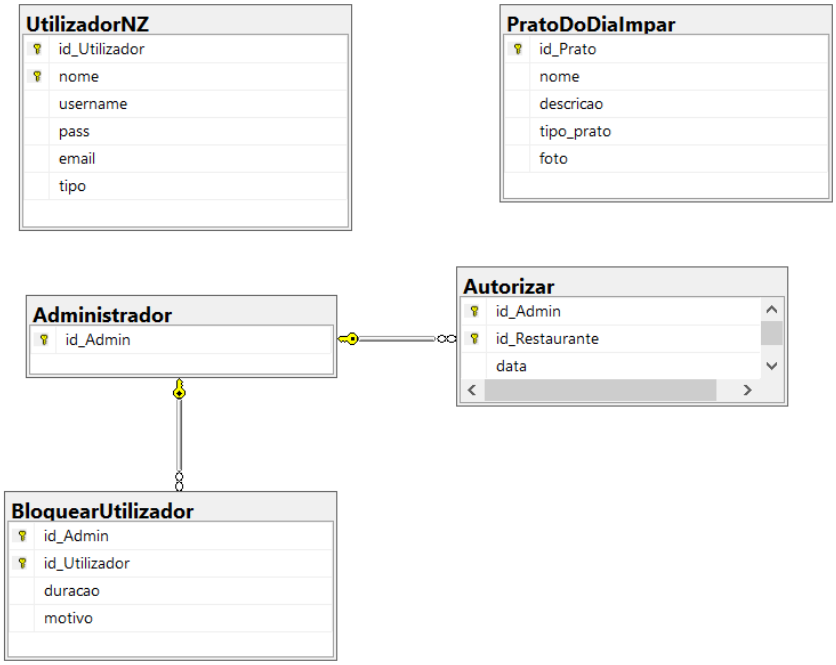

Figura 2: Diagrama servidor 2

Para o funcionamento da distribuição da base de dados original é necessário fazer algumas alterações na mesma, nomeadamente:

- A tabela não pode ter Identity.

-O campo de separação tem de pertencer à chave primária.

-A tabela de repartição não pode ter chaves estrangeiras.

-Os valores de default tem de ser atribuídos.

Divisão da base de dados distribuída:

De forma a elaborarmos uma base de dados distribuída, dividimos entre os dois servidores as tabelas “Prato_do_Dia” e “Utilizadores”. No servidor 1 foram colocadas todas as tabelas relacionadas com o cliente e restaurante, juntamente com as tabelas UtilizadorAM e PratoDoDiaPar. No servidor 2 restaram todas as tabelas relacionadas com o administrador e as tabelas repartidas referidas anteriormente (UtilizadorNZ, PratoDoDiaImpar).

Na tabela Prato_do_Dia decidimos dividir a tabela consoante a paridade do id_Prato, se este for par irá preencher no servidor 1, caso contrário, irá para o servidor 2. Na tabela utilizadores dividimos através da letra inicial do nome. Foram criadas também as 5 views e as respectivas permissões para os diversos utilizadores da base de dados distruibuida, necessárias para se poder aceder às tabelas do servidor 2.

Conclusão

Com base no que foi apresentado ao longo deste trabalho prático foi possível, com base na abordagem Top-Down criar um sistema homogéneo, capaz de fornecer aos utilizadores desta aplicação um acesso aos seus conteúdos de uma maneira mais rápida e organizada e, posto isto, pudemos melhorar as nossas competências no que diz respeito à criação, manutenção, proteção e funcionamento de uma base de dados distribuída.

Anexos

//Servidor 1

```
--Trabalho realizado por Vítor Neto 68717, João Leal 68719, Hugo Anes 68571 e
Leandro Coelho 68541--
---Server 1---

--Server 2 (Leandro)--
EXEC sp_addlinkedserver @server = 'Server2',
   @srvproduct = 'SQLServer Native Client OLEDB Provider',
   @provider = 'SQLNCLI',
   @datasrc = '25.80.186.225' --Endereço IP do 'server2'

EXEC sp_addlinkedsrvlogin @rmtsrvname = 'Server2',
   @useself = 'FALSE',
   @locallogin = 'sa',
   @rmtuser = 'sa',
   @rmtpassword = '12345'

USE master
GO
CREATE DATABASE TABD_TP2_SERVER1
USE TABD_TP2_SERVER1
GO

SET IMPLICIT_TRANSACTIONS OFF
SET XACT_ABORT ON
--Base de dados server 1-----------------------------------------------------------
------------------------------------------------------------------------------------
---------------------------------------------------------------------
--Nos 2 servers
CREATE TABLE UtilizadorAM(
       id_Utilizador INTEGER  ,
       nome varchar(50) ,
       username VARCHAR(50) UNIQUE NOT NULL,
       pass VARCHAR(50) NOT NULL,
       email VARCHAR(50) NOT NULL,
       tipo INTEGER,
       PRIMARY KEY(id_Utilizador, nome),
       CHECK(nome < 'N')
);

--Só no server 1
CREATE TABLE Restaurante(
       id_Restaurante INTEGER PRIMARY KEY,
       nome VARCHAR(50) NOT NULL,
       validado BIT DEFAULT 0 NOT NULL,
       morada VARCHAR(50) NOT NULL,
       gps VARCHAR(50) NOT NULL,
       telefone NUMERIC(9) UNIQUE NOT NULL,
       dia_descanso VARCHAR(50) NOT NULL,
       tipo_takeaway BIT DEFAULT 0,
       tipo_local BIT DEFAULT 0,
       tipo_entrega BIT DEFAULT 0,
       hora_de_abertura CHAR(5) NOT NULL,
       hora_de_fecho CHAR(5) NOT NULL,
       descricao VARCHAR(250),
       foto VARCHAR(300),
       --CHECK (hora_de_abertura LIKE '[0-1][0-9]:[0-5][0-9]' OR
       hora_de_abertura Like '[2][0-4]:[0-5][0-9]'),
```

```
	--CHECK (hora_de_fecho LIKE '[0-1][0-9]:[0-5][0-9]' OR hora_de_fecho Like
'[2][0-4]:[0-5][0-9]'),
	--CHECK (telefone BETWEEN 100000000 AND 999999999),
	--CHECK (tipo_takeaway LIKE '1' OR tipo_local LIKE '1' OR tipo_entrega
LIKE '1')
);

--Nos dois servers
CREATE TABLE PratoDoDiaPar(
	id_Prato INTEGER PRIMARY KEY,
	nome VARCHAR(250) NOT NULL,
	descricao VARCHAR(250),
	tipo_prato VARCHAR(250) NOT NULL,
	foto VARCHAR(300) NOT NULL,
	--CHECK(tipo_prato LIKE 'Vegan' OR tipo_prato LIKE 'Peixe' or tipo_prato
LIKE 'Carne'),
	CHECK(id_Prato%2 = 0)
);

--Só no server 1
CREATE TABLE CriarPrato(
	id_Restaurante INTEGER REFERENCES Restaurante(id_Restaurante) NOT NULL,
	id_Prato INTEGER,
	preco MONEY NOT NULL,
	dia_semana_prato VARCHAR(30) NOT NULL,
	apagado BIT NOT NULL DEFAULT 0, --0 não está apagado, 1 está
	PRIMARY KEY(id_Restaurante, id_Prato),
	--CHECK(dia_semana_prato LIKE 'Segunda feira' OR dia_semana_prato LIKE
'Terça feira' OR dia_semana_prato LIKE 'Quarta feira' or dia_semana_prato LIKE
'Quinta feira' OR dia_semana_prato LIKE 'Sexta feira' OR dia_semana_prato LIKE
'Sabado' OR dia_semana_prato LIKE 'Domingo')
	);

--Só no server 1
CREATE TABLE Client(
    id_Client INTEGER PRIMARY KEY,
    email_Confirmado BIT DEFAULT 0,
);

--Só no server 1
CREATE TABLE AdicionarPratoFav(
    id_Client INTEGER REFERENCES Client(id_Client) NOT NULL,
    id_Prato INTEGER NOT NULL,
    data_adicao DATE NOT NULL,
    PRIMARY KEY(id_Client, id_Prato),
);

--Só no server 1
CREATE TABLE AdicionarRestFav(
    id_Restaurante integer REFERENCES Restaurante(id_Restaurante) not null,
    id_Cliente integer REFERENCES Client(id_Client) not null,
    data_adicao DATE NOT NULL,
    Primary key(id_Restaurante, id_Cliente),
);

--Views-------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------------------------------------------------
CREATE VIEW PratoDoDia
AS
SELECT * FROM PratoDoDiaPar
UNION ALL
```

```
SELECT * FROM Server2.TABD_TP2_SERVER2.dbo.PratoDoDiaImpar
GO

CREATE VIEW Utilizador
AS
SELECT * FROM UtilizadorAM
UNION ALL
SELECT * FROM Server2.TABD_TP2_SERVER2.dbo.UtilizadorNZ
GO

CREATE VIEW BloquearUtilizador
AS
SELECT * FROM Server2.TABD_TP2_Server2.dbo.BloquearUtilizador
GO

CREATE VIEW Autorizar
AS
SELECT * FROM Server2.TABD_TP2_Server2.dbo.Autorizar
GO

CREATE VIEW Administrador
AS
SELECT * FROM Server2.TABD_TP2_Server2.dbo.Administrador
GO

--Logins-----------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------
CREATE LOGIN Administrador WITH PASSWORD = '1'
CREATE LOGIN Client WITH PASSWORD = '1'
CREATE LOGIN Restaurante WITH PASSWORD = '1'
CREATE LOGIN Dono WITH PASSWORD = '1'
GO

EXEC sp_addlinkedsrvlogin @rmtsrvname = 'Server2', @useself = 'TRUE' --Com o
comando seguinte, os logins acabados de criar, conseguem-se conectar ao servidor
2 com a mesma autenticação do servidor 1
USE TABD_TP2_SERVER1
GO

--Users------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------
CREATE USER Restaurante1 FOR LOGIN Restaurante
CREATE USER Administrador1 FOR LOGIN Administrador
CREATE USER Client1 FOR LOGIN Client
CREATE USER Dono FOR LOGIN Dono
CREATE USER Visitante WITHOUT LOGIN

--Atribuição de Permissões nas Tabelas----------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------
--Permissões para dono BD
GRANT SELECT, INSERT, UPDATE, DELETE TO Dono

--Permissões para os Restaurantes
GRANT SELECT, UPDATE ON Utilizador to Restaurante1
GRANT SELECT, INSERT, UPDATE, DELETE ON CriarPrato TO Restaurante1
GRANT SELECT, INSERT, UPDATE, DELETE ON PratoDoDia TO Restaurante1
GRANT SELECT, UPDATE ON Restaurante TO Restaurante1
GRANT EXECUTE ON Criar_Prato TO Restaurante1
GRANT EXECUTE ON Criar_Prato_Do_Dia TO Restaurante1
```

```
GRANT EXECUTE ON Alterar_Restaurante TO Restaurante1
GRANT EXECUTE ON Eliminar_Pratos_Restaurante TO Restaurante1
GRANT EXECUTE ON Eliminar_Pratos_Do_Dia_Restaurante TO Restaurante1
GRANT EXECUTE ON Alterar_Pratos_Restaurante TO Restaurante1

--Permissões para os Administradores
GRANT SELECT, DELETE ON Utilizador to Administrador1
GRANT SELECT, INSERT, UPDATE ON Administrador TO Administrador1
GRANT SELECT, DELETE ON Restaurante TO Administrador1
GRANT SELECT on CriarPrato TO Administrador1
GRANT SELECT on PratoDoDia TO Administrador1
GRANT EXECUTE ON Alterar_ClienteouAdmin TO Administrador1
GRANT EXECUTE ON Bloquear_Utilizador TO Administrador1
GRANT EXECUTE ON Autorizar_Restaurante TO Administrador1
GRANT EXECUTE ON Confirmar_Email TO Administrador1
GRANT EXECUTE ON Eliminar_Cliente TO Administrador1
GRANT EXECUTE ON Eliminar_Admin TO Administrador1
GRANT EXECUTE ON Eliminar_Restaurante TO Administrador1

--Permissões para os Clientes
GRANT SELECT, UPDATE ON Utilizador to Client1
GRANT SELECT ON PratoDoDia TO Client1
GRANT SELECT ON Restaurante TO Client1
GRANT EXECUTE ON Alterar_ClienteouAdmin TO Client1
GRANT EXECUTE ON Eliminar_Restaurante_Fav TO Client1
GRANT EXECUTE ON Eliminar_Prato_Fav TO Client1
GRANT EXECUTE ON Adicionar_PratoFavorito TO Client1
GRANT EXECUTE ON Adicionar_RestauranteFavorito TO Client1

--Permissões para Visitante
GRANT SELECT ON Restaurante TO Visitante
GRANT SELECT ON CriarPrato TO Visitante
GRANT SELECT ON PratoDoDia TO Visitante
GRANT EXECUTE ON Novo_ClienteOuAdmin TO Visitante
GRANT EXECUTE ON Novo_Restaurante TO Visitante

--Procedures Server 1-----------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------------------------------------------
--Procedures Necessários aos Utilizadores/Visitantes----------------------------
-------------------------------------------------------------------------------
-----------------------------------------------------------
CREATE procedure Novo_ClienteOuAdmin
      @nome varchar(50),@username varchar(50),@pass varchar(50), @email
varchar(50),@tipo integer
as
set transaction isolation level read committed

begin distributed transaction
      declare @IdMaisRecente INT
      select @IdMaisRecente = MAX(Utilizador.id_Utilizador) from Utilizador
      SET @IdMaisRecente = @IdMaisRecente + 1

      declare @testarTipo INT
      SET @testarTipo = @tipo

      if(@testarTipo = 0)
      begin
                  insert into Utilizador
                  values (@IdMaisRecente,@nome,@username,@pass,@email,0)
                  insert into Administrador
                  values(@IdMaisRecente)
      end
```

```
	if(@testarTipo = 1)
		begin
			insert into Utilizador
			values (@IdMaisRecente,@nome,@username,@pass,@email,1)
			insert into Client
			values(@IdMaisRecente, 0)
		end

	--if(@tipo = 2)
	--insert into Utilizador
	--values (@IdMaisRecente,@username,@pass,@email,2)
	--insert into Restaurante
	--values(@IdMaisRecente)

	if(@@ERROR <> 0) OR (@@ROWCOUNT = 0)
	GOTO ERRO
	COMMIT
return 1
	ERRO:
		ROLLBACK
		RETURN -1
GO
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------------------------------------------------
CREATE PROCEDURE Novo_Restaurante
	@nome varchar(20), @morada VARCHAR(50), @gps VARCHAR(50), @telefone
NUMERIC(9), @dia_desc VARCHAR(50), @tt BIT, @tl BIT, @te BIT, @hora_aber CHAR(5),
@hora_fecho CHAR(5), @desc VARCHAR(250), @foto VARCHAR(300),
	@username varchar(50), @password varchar(50), @email varchar(50)
as
set transaction isolation level read committed
begin DISTRIBUTED transaction
	declare @IdMaisRecente INT
	select @IdMaisRecente = MAX(id_Utilizador) from Utilizador
	insert into Restaurante
	values((@IdMaisRecente + 1),@nome,0, @morada, @gps, @telefone, @dia_desc,
@tt, @tl, @te, @hora_aber, @hora_fecho, @desc, @foto)

	INSERT INTO Utilizador
	VALUES( (@IdMaisRecente + 1), @nome, @username, @password, @email, 2)

	if(@@ERROR <> 0) OR (@@ROWCOUNT = 0)
	GOTO ERRO
	COMMIT
return 1
	ERRO:
		ROLLBACK
		RETURN -1
GO
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------------------------------------------------
CREATE PROCEDURE Criar_Prato
	@id_R integer, @id_P integer, @preco money, @dia_semana varchar(30),
@apagado bit
as
BEGIN DISTRIBUTED TRANSACTION
	insert into CriarPrato
	values (@id_R,@id_P,@preco,@dia_semana,'0')
	IF (@@ERROR <> 0) OR (@@ROWCOUNT = 0)
```

```
		GOTO ERRO
		COMMIT
return 1
	ERRO:
		ROLLBACK
		RETURN -1

GO

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------------------------------------------------
CREATE PROCEDURE Criar_Prato_Do_Dia
	@id_P integer, @nome varchar(25), @descricao varchar(200),@tipo_prato
varchar(20),@foto varchar(300)
as
BEGIN DISTRIBUTED TRANSACTION
	insert into PratoDoDia(id_Prato,nome,descricao,tipo_prato,foto)
	values (@id_P,@nome,@descricao,@tipo_prato,@foto)
	IF (@@ERROR <> 0) OR (@@ROWCOUNT = 0)
		GOTO ERRO
		COMMIT
return 1
	ERRO:
		ROLLBACK
		RETURN -1
GO
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------------------------------------------------
CREATE PROCEDURE Alterar_Restaurante
	 @id_Utilizador INTEGER, @nome varchar(50), @username varchar(50),
@password varchar(50), @email varchar(50), @morada varchar(50), @gps VARCHAR(50),
@telefone NUMERIC(9), @dia_desc VARCHAR(50), @tt BIT, @tl BIT, @te BIT,
@hora_aber CHAR(5), @hora_fecho CHAR(5), @desc VARCHAR(250), @foto VARCHAR(300)
as
	set transaction isolation level read committed
	begin transaction
	update Restaurante
	set  nome = @nome, morada = @morada, gps = @gps, telefone = @telefone,
dia_descanso = @dia_desc, tipo_takeaway = @tt, tipo_local = @tl, tipo_entrega =
@te, hora_de_abertura = @hora_aber, hora_de_fecho = @hora_fecho, descricao =
@desc, foto = @foto
	where id_Restaurante = @id_Utilizador

	update Utilizador
	set id_Utilizador=@id_Utilizador, nome=@nome, username= @username,
pass=@password, email=@email, tipo=2

	if(@@ERROR <> 0) OR (@@ROWCOUNT = 0)
	GOTO ERRO
	COMMIT
return 1
	ERRO:
		ROLLBACK
		RETURN -1
GO
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------------------------------------------------
CREATE PROCEDURE Alterar_ClienteouAdmin
	@nome varchar(50), @username varchar(50), @password varchar(50), @email
varchar(50)
```

```
as
	set transaction isolation level read committed
	begin transaction
	declare @IdMaisRecente INT
	select @IdMaisRecente = MAX(id_Utilizador) from Utilizador

	update Cliente
	set id_Client = @IdMaisRecente

	update Administrador
	set id_Admin = @IdMaisRecente

	update Utilizador
	set nome=@nome, username= @username, pass=@password, email=@email

	if(@@ERROR <> 0) OR (@@ROWCOUNT = 0)
	GOTO ERRO
	COMMIT
return 1
	ERRO:
		ROLLBACK
		RETURN -1
GO
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------
CREATE PROCEDURE Eliminar_Restaurante_Fav
	 @id_Cliente integer,@id_Restaurante integer
as
set transaction isolation level repeatable read
begin transaction
    delete from AdicionarRestFav
    where(id_Cliente=@id_Cliente and id_Restaurante=@id_Restaurante)
    if(@@ERROR <> 0) OR (@@ROWCOUNT = 0)
    GOTO ERRO
    COMMIT
return 1
    ERRO:
        ROLLBACK
        RETURN -1
GO
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------
CREATE PROCEDURE Eliminar_Prato_Fav
       @id_Cliente integer,@id_Prato integer
as
set transaction isolation level repeatable read
begin transaction
    delete from AdicionarPratoFav
    where(id_Client=@id_Cliente and id_Prato=@id_Prato)
    if(@@ERROR <> 0) OR (@@ROWCOUNT = 0)
    GOTO ERRO
    COMMIT
return 1
    ERRO:
        ROLLBACK
        RETURN -1
GO
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------
```

```
CREATE PROCEDURE Eliminar_Pratos_Restaurante
    @id_Restaurante integer, @id_Prato integer
as
set transaction isolation level repeatable read
begin transaction

       delete from AdicionarPratoFav
    where(id_Prato=@id_Prato)
    delete from CriarPrato
    where(id_Restaurante=@id_Restaurante and id_Prato=@id_Prato)
    delete from PratoDoDia
    where(id_Prato=@id_Prato)

    if(@@ERROR <> 0) OR (@@ROWCOUNT = 0)
    GOTO ERRO
    COMMIT
return 1
    ERRO:
        ROLLBACK
        RETURN -1
GO
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------
CREATE PROCEDURE Eliminar_Pratos_Do_Dia_Restaurante
    @id_Restaurante integer, @id_Prato integer, @dia_da_semana varchar
as
set transaction isolation level repeatable read
begin distributed transaction
    delete from AdicionarPratoFav
    where(id_Prato=@id_Prato)
    delete from CriarPrato
    where(id_Restaurante=@id_Restaurante and dia_semana_prato=@dia_da_semana)
    delete from PratoDoDia
    where(id_Prato=@id_Prato)

    if(@@ERROR <> 0) OR (@@ROWCOUNT = 0)
    GOTO ERRO
    COMMIT
return 1
    ERRO:
        ROLLBACK
        RETURN -1
GO-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------
CREATE PROCEDURE Bloquear_Utilizador
       @id_Admin integer, @id_Utilizador integer, @duracao integer, @motivo
varchar
as
set transaction isolation level read committed
begin distributed transaction
       insert into BloquearUtilizador(id_Admin,id_Utilizador,duracao,motivo)
       values (@id_Admin,@id_Utilizador,@duracao,@motivo)
       if(@@ERROR <> 0) OR (@@ROWCOUNT = 0)
       GOTO ERRO
       COMMIT
return 1
       ERRO:
              ROLLBACK
              RETURN -1
GO
```

```
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------
CREATE PROCEDURE Alterar_Pratos_Restaurante
	@id_Restaurante integer, @id_Prato integer, @nome Varchar(50), @preco
money,@dia_semana_prato varchar(50), @descricao varchar(250), @tipo_prato
varchar(50)
as
set transaction isolation level repeatable read
begin transaction
	update CriarPrato
	set id_Prato=@id_Prato,preco=@preco,dia_semana_prato=@dia_semana_prato
	where id_Restaurante=@id_Restaurante
	update PratoDoDia
	set nome=@nome, descricao=@descricao, tipo_prato=@tipo_prato
	where id_Prato=@id_Prato
	if(@@ERROR <> 0) OR (@@ROWCOUNT = 0)
	GOTO ERRO
	COMMIT
return 1
	ERRO:
		ROLLBACK
		RETURN -1
GO
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------
CREATE PROCEDURE Confirmar_Email
	@id_Utilizador varchar(50)
as
set transaction isolation level read committed
begin transaction
	update Client
	set Email_Confirmado=1
	where id_Client=@id_Utilizador
	IF (@@ERROR <> 0) OR (@@ROWCOUNT = 0)
		GOTO ERRO
COMMIT
RETURN 1

ERRO:
	ROLLBACK
	RETURN -1
GO
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------
CREATE PROCEDURE Autorizar_Restaurante
	@id_A integer, @id_R integer
as
set transaction isolation level read committed
begin transaction
	insert into Autorizar(id_Admin,id_Restaurante,data)
	values (@id_A,@id_R,GETDATE())
	UPDATE Restaurante
	SET validado = 1
	WHERE @id_R = id_Restaurante
	IF (@@ERROR <> 0) OR (@@ROWCOUNT = 0)
		GOTO ERRO
COMMIT
RETURN 1
```

```
ERRO:
	ROLLBACK
	RETURN -1
GO
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------
CREATE PROCEDURE Adicionar_PratoFavorito
	@ID_C INTEGER,
	@ID_P INTEGER
AS
BEGIN DISTRIBUTED TRANSACTION
	INSERT INTO AdicionarPratoFav(id_Client,id_Prato,data_adicao)
	VALUES (@ID_C,@ID_P,GETDATE())
	IF (@@ERROR <> 0) OR (@@ROWCOUNT = 0)
		GOTO ERRO
COMMIT
RETURN 1

ERRO:
	ROLLBACK
	RETURN -1
GO
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------
CREATE PROCEDURE Adicionar_RestauranteFavorito
	@ID_C INTEGER,
	@ID_R INTEGER
AS
BEGIN DISTRIBUTED TRANSACTION
	INSERT INTO AdicionarRestFav(id_Cliente,id_Restaurante,data_adicao)
	VALUES (@ID_C,@ID_R,GETDATE())
	IF (@@ERROR <> 0) OR (@@ROWCOUNT = 0)
		GOTO ERRO
		COMMIT
RETURN 1

ERRO:
	ROLLBACK
	RETURN -1
GO
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------
CREATE PROCEDURE Eliminar_Cliente
    @id_Cliente integer
as
set transaction isolation level read committed
begin transaction
    delete from AdicionarPratoFav
    where(id_Client=@id_Cliente)
    delete from AdicionarRestFav
    where(id_Cliente=@id_Cliente)
    delete from Cliente
    where(id_Client=@id_Cliente)
       delete from Utilizador
       where(id_Utilizador=@id_Cliente)
    if(@@ERROR <> 0) OR (@@ROWCOUNT = 0)
    GOTO ERRO
    COMMIT
return 1
```

```
    ERRO:
        ROLLBACK
        RETURN -1
GO
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------
CREATE PROCEDURE Eliminar_Admin
    @id_Admin integer
as
set transaction isolation level read committed
begin DISTRIBUTED transaction
       delete from Administrador
    where(id_Admin=@id_Admin)
    delete from BloquearUtilizador
    where(id_Admin=@id_Admin)
       delete from Utilizador
       where(id_Utilizador=@id_Admin)
    if(@@ERROR <> 0) OR (@@ROWCOUNT = 0)
    GOTO ERRO
    COMMIT
return 1
    ERRO:
        ROLLBACK
        RETURN -1
GO
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------
CREATE PROCEDURE Eliminar_Restaurante
    @id_Restaurante integer
as
set transaction isolation level read committed
begin distributed transaction
    delete from AdicionarRestFav
    where(id_Restaurante=@id_Restaurante)
    delete from Restaurante
    where(id_Restaurante=@id_Restaurante)
       delete from Utilizador
       where(id_Utilizador=@id_Restaurante)
    if(@@ERROR <> 0) OR (@@ROWCOUNT = 0)
    GOTO ERRO
    COMMIT
return 1
    ERRO:
        ROLLBACK
        RETURN -1
GO
```

```
//Servidor 2

--Trabalho realizado por Vítor Neto 68717, João Leal 68719, Hugo Anes 68571 e
Leandro Coelho 68541--
--Server 2---

--Ligar com o server1 (Vítor)
EXEC sp_addlinkedserver @server = 'server1',
   @srvproduct = 'SQLServer Native Client OLEDB Provider',
   @provider = 'SQLNCLI',
   @datasrc = '25.73.248.232' --Endereco IP do 'server1'

EXEC sp_addlinkedsrvlogin @rmtsrvname = 'server1',
   @useself = 'FALSE',
   @locallogin = 'sa',
   @rmtuser = 'sa',
   @rmtpassword = 'admintabd'

CREATE DATABASE TABD_TP2_SERVER2
USE TABD_TP2_SERVER2
GO
SET IMPLICIT_TRANSACTIONS OFF

--Base de dados server 2-----------------------------------------------------------
------------------------------------------------------------------------------------
-------------------------------------------------------------------
--Nos 2 servers
CREATE TABLE UtilizadorNZ(
       id_Utilizador INTEGER ,
       nome VARCHAR(50) NOT NULL,
       username VARCHAR(50) UNIQUE NOT NULL,
       pass VARCHAR(50) NOT NULL,
       email VARCHAR(50) NOT NULL,
       tipo INTEGER,
       PRIMARY KEY (id_Utilizador,nome),
       CHECK(tipo LIKE '0' OR tipo LIKE '1' OR tipo LIKE '2'), --0 = Admin, 1 =
Client, 2 = Restaurante
       CHECK(email LIKE '%@%.%'),
       CHECK(nome >= 'N')
);

--Nos 2 servers
CREATE TABLE PratoDoDiaImpar(
       id_Prato INTEGER PRIMARY KEY,
       nome VARCHAR(250) NOT NULL,
       descricao VARCHAR(250),
       tipo_prato VARCHAR(250) NOT NULL,
       foto VARCHAR(300) NOT NULL,
       CHECK(tipo_prato LIKE 'Vegan' OR tipo_prato LIKE 'Peixe' or tipo_prato
LIKE 'Carne'),
       CHECK(id_Prato%2 <> 0)
);

--Só no server 2
CREATE TABLE Administrador(
    id_Admin INTEGER Primary Key,
);

--Só no server 2
CREATE TABLE BloquearUtilizador(
       id_Admin INTEGER,
       id_Utilizador INTEGER,
       duracao INTEGER NOT NULL,
```

```
	motivo VARCHAR(250) NOT NULL,
	PRIMARY KEY(id_Admin,id_Utilizador),
	FOREIGN KEY (id_Admin) REFERENCES Administrador(id_Admin)
);

--Só no server 2
CREATE TABLE Autorizar(
	id_Admin INTEGER,
	id_Restaurante INTEGER,
	data DATETIME NOT NULL DEFAULT GETDATE(),
	PRIMARY KEY(id_Admin, id_Restaurante),
	FOREIGN KEY (id_Admin) REFERENCES Administrador(id_Admin)
);

--Logins-----------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------
CREATE LOGIN Administrador WITH PASSWORD = '1'
CREATE LOGIN Client WITH PASSWORD = '1'
CREATE LOGIN Restaurante WITH PASSWORD = '1'
CREATE LOGIN Dono WITH PASSWORD = '1'

--Users--
CREATE USER Restaurante1 FOR LOGIN Restaurante
CREATE USER Administrador1 FOR LOGIN Administrador
CREATE USER Client1 FOR LOGIN Client
CREATE USER Dono FOR LOGIN Dono
CREATE USER Visitante WITHOUT LOGIN

--Atribuição de Permissões nas Tabelas----------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------
--Permissões para dono BD
GRANT SELECT, INSERT, UPDATE, DELETE TO Dono

--Permissões para os Restaurantes
GRANT SELECT, UPDATE ON UtilizadorNZ to Restaurante1

--Permissões para os Administradores
GRANT SELECT, DELETE ON UtilizadorNZ to Administrador1
GRANT SELECT, DELETE ON Client TO Administrador1
GRANT SELECT, INSERT, UPDATE, DELETE on Autorizar TO Administrador1
GRANT SELECT, UPDATE on BloquearUtilizador TO Administrador1

--Permissões para os Clientes
GRANT SELECT, UPDATE ON UtilizadorNZ to Cliente1
GRANT SELECT ON PratoDoDiaImpar TO Cliente1
GRANT SELECT, UPDATE ON Client TO Client1
GRANT SELECT, INSERT, UPDATE, DELETE ON AdicionarRestFav to Client1
GRANT SELECT, INSERT, UPDATE, DELETE ON AdicionarPratoFav to Client1

--Permissões para Visitante
GRANT SELECT ON PratoDoDiaImpar TO Visitante
```