Instituto Superior de Engenharia de Lisboa
Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores

Mestrado em Engenharia Informática e de Computadores

Arquitectura de Sistemas Distribuidos

Relatório Aula Prática Nº 1

Nome	Nº de Aluno	e-mail
Rui Miranda	A32342	a32342@alunos.isel.pt
David Coelho	A21359	a21359@alunos.isel.pt
Frederico Ferreira	A7066	a7066@alunos.isel.pt

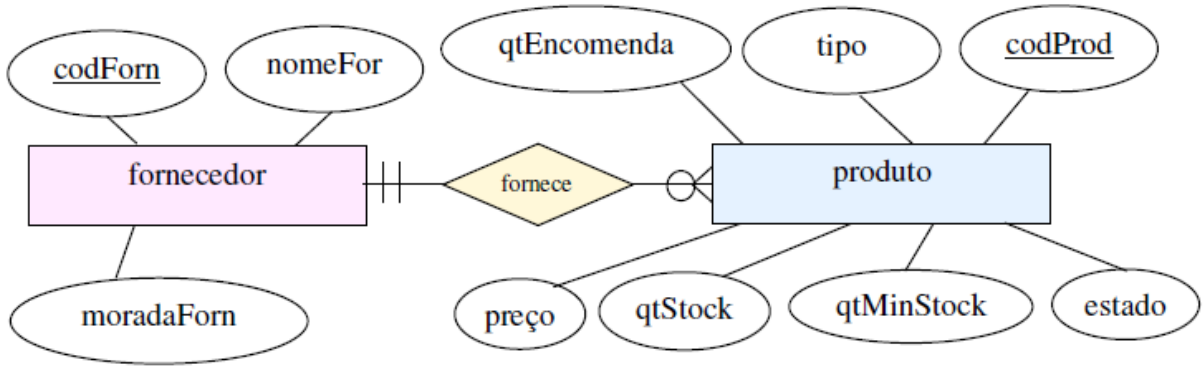
DEETC, Outubro de 2013

Sumário

O presente documento descreve a realização de um trabalho prático ligado à implementação de bases de dados distribuidas e que teve por base o enunciado publicado.
A tecnologia usada foi SQL Server 2012 e o ambiente distribuido simulado com 3 instâncias independentes a correr na mesma máquina.
A actividade permitiu usar os conceitos teóricos estudados nas aulas teóricas, incluindo a construção dos Esquema Lógico Global, Esquema de Fragementação e do Esquema de Distribuição.
Foram ainda usados os mecanismos disponíveis de Vistas, Sinónimos, Procedimentos e "Merge replication".

A prática tem por base a impletação da base de dados de suporte a uma empresa fictícia de comercialização de artigos de desporto e de criança, que distribui a sua actividade por uma sede e duas lojas e que tem alguns requisitos de operação, que impõe a distribuição dos dados.
Os requisitos iniciais (alterados ao longo dos exercícios orientadores da aula prática são):
1. As aquisições as fornecedores são realizadas pela sede, uma vez por semana, sendo para cada produto com quantidade em stock inferior ao stock mínimo, encomendada uma quantidade dependente do produto. Cada produto é vendido por um único fornecedo;
2. As vendas apenas são realizadas nos centros de vendas, sendo aí relevante a informação sobre o produto e a quantidade em stock;
3. Deve garantir-se que os centros de venda funcionam mesmo na ausência de comunicações com a sede e sem necessidade de comunicarem um com o outro;
4. Na sede deve ser garantida a visão global da base de dados.

A figura abaixo, corresponde ao modleo E/A correspondente ao esquema lógico global:



Exercício 1

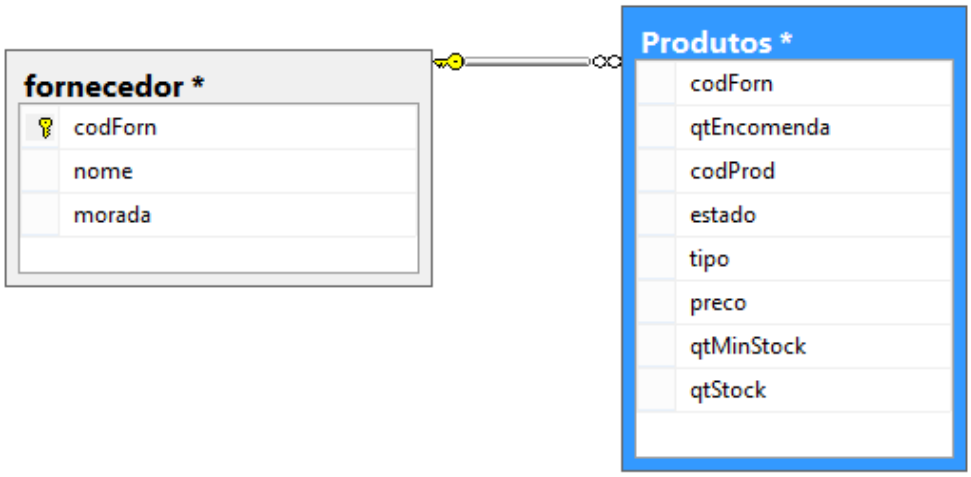
a) Conceba os esquemas lógico global e de fragmentação e distribua a base de dados por três instâncias do Sql Server, de acordo com o esquema de fragmentação adoptado.
b) Sabendo que o único local onde se pretende ter acesso global aos dados é na sede, desenvolva uma solução que apresente níveis adequados de independência do esquema lógico global relativamente ao esquema de fragmentação e deste relativamente ao esquema de distribuição, sabendo que as inserções e remoções de produtos podem ser realizadas através de procedimentos armazenados, mas que as actualizações de produtos devem poder ser realizadas através da execução directa de instruções UPDATE.
c) Construa um procedimento armazenado para ser usado na sede que permita lançar as encomendas dos produtos vendidos em ambos os centros de vendas.

Resolução

Esquema Lógico Global da Base de Dados

Esquema Logico Global

Produto(codProd, codForn [FK],tipo,estado,preço,qtStock,qtMinStock,qtEncomenda)
Fornecedor(codForn, nomeForn, moradaForn)



Esquema de Fragmentação pelas 3 Bases de Dados

Vertical entre a Sede e as Lojas e Horizontal entre lojas tendo por o tipo de produto comercializado em cada Loja.

Esquema de Fragmentação

Partição vertical (centros de vendas e lojas)

Produto(codProd, codForn [FK], tipo, preço, qtStock, qtMinStock, qtEncomenda)

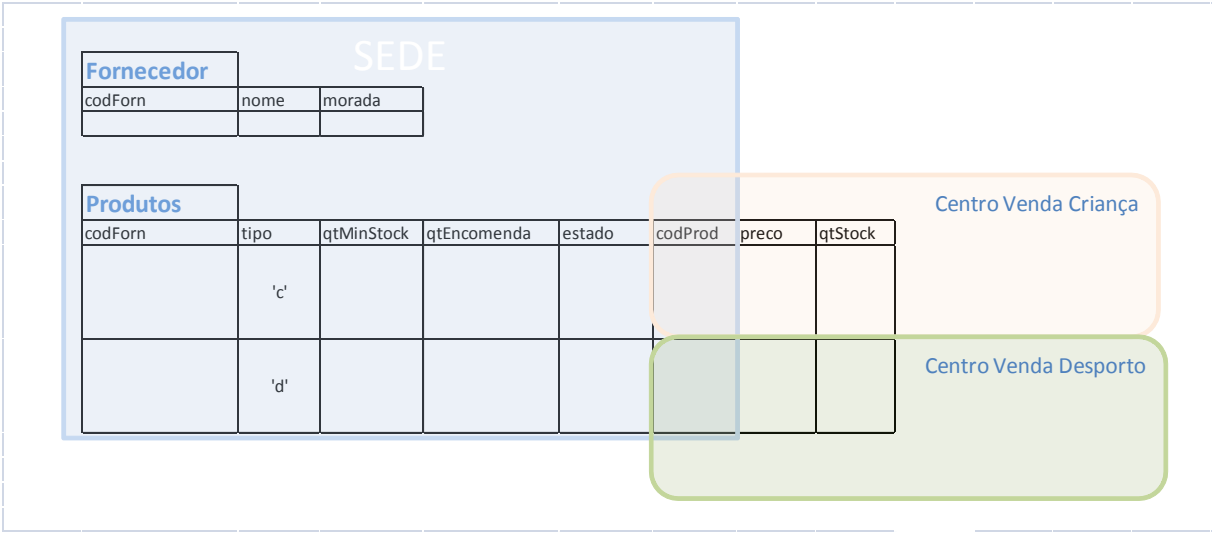
- ProdutoSede(codProd,codForn [FK] ,estado,qtMinStock,qtEncomenda)
  - π\{codProd,CodForn,qtMinStock,qtEncomenda\}(Produto)
- ProdutoLojas(codProd, tipo, preço, qtStock)
  - π\{codProd,tipo,qtStock\}(Produto)

Partição Horizontal (centros de vendas)

ProdutoLojas(codProd, tipo, preço, qtStock)

- ProdutoLojaDesp = σ\{tipo = 'desporto'\}(ProdutoLojas)
- ProdutoLojaCriança = σ\{tipo = 'criança'\}(ProdutoLojas)

De um ponto de vista conceptual o tipo de produto deveria estar nas lojas, contudo na implementação por uma questão de desempenho (para reduzir acessos distribuídos quando não necessários) foi colocado na Sede como representado na figura.



Script de criação da base de dados da SEDE

```
USE [master]
GO

CREATE DATABASE [ASI]
GO

USE [ASI]
GO
/****** Object:  Table [dbo].[fornecedor]    Script Date: 17-10-2013 20:54:06 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[fornecedor](
	[cod] [int] NOT NULL,
	[nome] [varchar](20) NULL,
	[morada] [varchar](60) NULL,
PRIMARY KEY CLUSTERED
(
	[cod] 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
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[produto]    Script Date: 17-10-2013 20:54:06 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[produto](
	[codFornecedor] [int] NULL,
	[qtEncomenda] [int] NULL,
	[cod] [int] NOT NULL,
```

```
	[estado] [bit] NULL,
	[tipo] [char](1) NOT NULL,
PRIMARY KEY CLUSTERED
(
	[cod] 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
SET ANSI_PADDING OFF
GO
ALTER TABLE [dbo].[produto]  WITH CHECK ADD  CONSTRAINT [fk_produto_fornecedor] FOREIGN KEY([codFornecedor])
REFERENCES [dbo].[fornecedor] ([cod])
GO
ALTER TABLE [dbo].[produto] CHECK CONSTRAINT [fk_produto_fornecedor]
GO
```

Script de criação da base de dados da Loja

```
USE [master]
GO

CREATE DATABASE [ASI]
GO

USE [ASI]
GO
/****** Object:  Table [dbo].[produto]    Script Date: 17-10-2013 21:00:36 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[produto](
	[cod] [int] NOT NULL,
	[preco] [money] NULL,
	[qtStock] [int] NULL,
	[qtMinStock] [int] NULL,
PRIMARY KEY CLUSTERED
(
	[cod] 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
```

Script de criação dos LinkedServers

```
USE [ASI]
GO

--Loja Crianças
/****** Object:  LinkedServer [MIRANDA-LAPTOP\SQL2012DEINST2]    Script Date: 17-10-2013 21:48:19 ******/
EXEC master.dbo.sp_dropserver @server=N'MIRANDA-LAPTOP\SQL2012DEINST2', @droplogins='droplogins'
GO

/****** Object:  LinkedServer [MIRANDA-LAPTOP\SQL2012DEINST2]    Script Date: 17-10-2013 21:48:19 ******/
EXEC master.dbo.sp_addlinkedserver @server = N'MIRANDA-LAPTOP\SQL2012DEINST2', @srvproduct=N'SQL Server'
 /* For security reasons the linked server remote logins password is changed with ######## */
```

```
EXEC master.dbo.sp_addlinkedsrvlogin @rmtsrvname=N'MIRANDA-
LAPTOP\SQL2012DEINST2',@useself=N'True',@locallogin=NULL,@rmtuser=NULL,@rmtpassword=NULL
GO

/****** Object:  Synonym [dbo].[ProdutoCrianca]    Script Date: 17-10-2013 21:47:39 ******/
DROP SYNONYM [dbo].[ProdutoCrianca]
GO

/****** Object:  Synonym [dbo].[ProdutoCrianca]    Script Date: 17-10-2013 21:47:39 ******/
CREATE SYNONYM [dbo].[ProdutoCrianca] FOR [MIRANDA-LAPTOP\SQL2012DEINST2].[ASI].[dbo].[Produto]
GO



--Loja Desportitas
/****** Object:  LinkedServer [MIRANDA-LAPTOP\SQL2012DEINST2]    Script Date: 17-10-2013 21:48:19 ******/
EXEC master.dbo.sp_dropserver @server=N'MIRANDA-LAPTOP\SQL2012DEINST3', @droplogins='droplogins'
GO

/****** Object:  LinkedServer [MIRANDA-LAPTOP\SQL2012DEINST2]    Script Date: 17-10-2013 21:48:19 ******/
EXEC master.dbo.sp_addlinkedserver @server = N'MIRANDA-LAPTOP\SQL2012DEINST3', @srvproduct=N'SQL Server'
 /* For security reasons the linked server remote logins password is changed with ######## */
EXEC master.dbo.sp_addlinkedsrvlogin @rmtsrvname=N'MIRANDA-
LAPTOP\SQL2012DEINST3',@useself=N'True',@locallogin=NULL,@rmtuser=NULL,@rmtpassword=NULL
GO

/****** Object:  Synonym [dbo].[ProdutoCrianca]    Script Date: 17-10-2013 21:47:39 ******/
DROP SYNONYM [dbo].[ProdutoDesp]
GO

/****** Object:  Synonym [dbo].[ProdutoCrianca]    Script Date: 17-10-2013 21:47:39 ******/
CREATE SYNONYM [dbo].[ProdutoDesp] FOR [MIRANDA-LAPTOP\SQL2012DEINST3].[ASI].[dbo].[Produto]
GO
```

Procedimento de Inserção de Produtos

```
USE [ASI]
GO

/****** Object:  StoredProcedure [dbo].[insereProduto]    Script Date: 24-10-2013 12:19:24 ******/
DROP PROCEDURE [dbo].[insereProduto]
GO

/****** Object:  StoredProcedure [dbo].[insereProduto]    Script Date: 24-10-2013 12:19:24 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO



create procedure [dbo].[insereProduto]
	@cod			[int],
	@codFornecedor[int],
	@qtEncomenda  [int],
	@estado				[bit],
	@tipo			[char](1),
	@preco			[money],
	@qtMinStock		[int],
	@qtStock		[int]
```

```
as
begin
	SET XACT_ABORT ON
	BEGIN TRANSACTION T_insereProduto

		INSERT INTO [dbo].[produto]
					([codFornecedor]
					,[qtEncomenda]
					,[cod]
					,[estado]
					,[tipo])
				VALUES
					(@codFornecedor
					,@qtEncomenda
					,@cod
					,@estado
					,@tipo)

		IF @tipo = 'D'
		BEGIN
			INSERT INTO ProdutoDesp
						([cod]
						,[preco]
						,[qtStock]
						,[qtMinStock])
					VALUES
						(@cod
						,@preco
						,@qtStock
						,@qtMinStock)

		END
		IF @tipo = 'C'
		BEGIN
			INSERT INTO ProdutoCrianca
						([cod]
						,[preco]
						,[qtStock]
						,[qtMinStock])
					VALUES
						(@cod
						,@preco
						,@qtStock
						,@qtMinStock)
		END



	COMMIT TRANSACTION T_insereProduto
	RETURN
end
GO
Sede: Vista Sobre Produtos
USE [ASI]
GO

/****** Object:  View [dbo].[viewProduto]    Script Date: 17-10-2013 21:20:07 ******/
DROP VIEW [dbo].[viewProduto]
GO
```

```
/****** Object:  View [dbo].[viewProduto]    Script Date: 17-10-2013 21:20:07 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE VIEW [dbo].[viewProduto]
AS

	SELECT
		lp.cod, lp.codFornecedor, lp.qtEncomenda, lp.estado, lp.tipo,
		rp.preco, rp.qtMinStock, rp.qtStock
	FROM
		dbo.produto lp
		inner join (
			select * from ProdutoCrianca union
			select * from ProdutoDesp
		) rp
			on lp.cod = rp.cod

GO
```

SEDE: trigger de Upgrade sobre a View de Produtos

```
USE [ASI]
GO

/****** Object:  Trigger [trViewProduto]    Script Date: 24-10-2013 12:33:44 ******/
DROP TRIGGER [dbo].[trViewProduto]
GO

/****** Object:  Trigger [dbo].[trViewProduto]    Script Date: 24-10-2013 12:33:45 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO


CREATE TRIGGER [dbo].[trViewProduto]
ON [dbo].[viewProduto]
INSTEAD OF UPDATE AS
begin
	SET XACT_ABORT ON
	BEGIN TRANSACTION T_trViewProduto

		UPDATE ProdutoCrianca
		SET
			ProdutoCrianca.preco = inserted.preco,
			ProdutoCrianca.qtMinStock = inserted.qtMinStock,
			ProdutoCrianca.qtMinStock = inserted.qtStock
		FROM
			ProdutoCrianca
		INNER JOIN
			inserted
		ON
			inserted.cod = ProdutoCrianca.cod
```

```
                UPDATE ProdutoDesp
                SET
                        ProdutoDesp.preco = inserted.preco,
                        ProdutoDesp.qtMinStock = inserted.qtMinStock,
                        ProdutoDesp.qtMinStock = inserted.qtStock
                FROM
                        ProdutoDesp
                INNER JOIN
                        inserted
                ON
                        inserted.cod = ProdutoDesp.cod

                UPDATE Produto
                SET
                        Produto.codFornecedor = inserted.codFornecedor,
                        Produto.estado = inserted.estado,
                        Produto.qtEncomenda = inserted.qtEncomenda
                FROM Produto
                INNER JOIN
                        inserted
                ON
                        inserted.cod = Produto.cod

        COMMIT TRANSACTION T_trViewProduto

end


GO
```

SEDE: Procedimento de Encomenda de Produtos

```
USE [ASI]
GO
/****** Object:  StoredProcedure [dbo].[produtosQueDevemSerEncomendados]    Script Date: 17-10-2013 22:21:18
******/
DROP PROCEDURE [dbo].[produtosQueDevemSerEncomendados]
GO

/****** Object:  StoredProcedure [dbo].[produtosQueDevemSerEncomendados]    Script Date: 17-10-2013 22:21:18
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

create procedure [dbo].[produtosQueDevemSerEncomendados] as
begin
        select f.nome, f.morada, p.cod, p.qtEncomenda
        from fornecedor f inner join produto p on f.cod = p.codFornecedor
        where 1=1
                and p.estado = 0
                and p.cod in (
                        select     p.cod
```

```
                        from    [ProdutoCrianca] p
                        where   p.qtStock < p.qtMinStock
                        union
                        select  p.cod
                        from    [ProdutoDesp] p
                        where   p.qtStock < p.qtMinStock
                )
        ;
end

GO

/****** Object:  StoredProcedure [dbo].[produtosEfectivamenteEncomendados]    Script Date: 17-10-2013 22:20:56
******/
DROP PROCEDURE [dbo].[produtosEfectivamenteEncomendados]
GO

/****** Object:  StoredProcedure [dbo].[produtosEfectivamenteEncomendados]    Script Date: 17-10-2013 22:20:56
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO


create procedure [dbo].[produtosEfectivamenteEncomendados] as
begin
        update produto
        set estado = 1
        where cod in
        (
                select  p.cod
                from    [ProdutoCrianca] p
                where   p.qtStock < p.qtMinStock
                union
                select  p.cod
                from    [ProdutoDesp] p
                where   p.qtStock < p.qtMinStock
        )
        ;
end

GO
/****** Object:  StoredProcedure [dbo].[produtosEncomendadosForamRecebidos]    Script Date: 17-10-2013 22:21:36
******/
DROP PROCEDURE [dbo].[produtosEncomendadosForamRecebidos]
GO

/****** Object:  StoredProcedure [dbo].[produtosEncomendadosForamRecebidos]    Script Date: 17-10-2013 22:21:36
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO


create procedure [dbo].[produtosEncomendadosForamRecebidos] as
begin
```

```
        UPDATE [dbo].[viewProduto]
        SET     qtStock = qtStock + qtEncomenda
        WHERE estado = 1

end

GO

/****** Object:  StoredProcedure [dbo].[produtosEncomendadosForamRecebidos]    Script Date: 17-10-2013 22:21:36
******/
DROP PROCEDURE [dbo].[produtosEncomendadas]
GO

/****** Object:  StoredProcedure [dbo].[produtosEncomendadosForamRecebidos]    Script Date: 17-10-2013 22:21:36
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO


create procedure [dbo].[produtosEncomendadas] as
begin
        SET XACT_ABORT ON
        BEGIN TRANSACTION T_produtosEncomendadas
                EXEC [dbo].[produtosQueDevemSerEncomendados]
                EXEC [dbo].[produtosEfectivamenteEncomendados]
                EXEC [dbo].[produtosEncomendadosForamRecebidos]
        COMMIT TRANSACTION T_produtosEncomendadas
end

GO
```

Procedimento de receção de Produtos

```
USE [ASI]
GO

DROP PROCEDURE [dbo].produtoEncomendadoFoiRecebido
GO

SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO


create procedure [dbo].produtoEncomendadoFoiRecebido
        @cod                    [int],
        @qtdEncomenda[int]
as
begin

        UPDATE [dbo].[viewProduto]
        SET     qtStock = qtStock + @qtdEncomenda
        WHERE cod=@cod AND estado=1
```

```
end

GO
```

Procedimento de Alteração de Tipo de Produto

```
USE [ASI]
GO

/****** Object:  StoredProcedure [dbo].[produtoAlteraTipo]    Script Date: 24-10-2013 12:19:24 ******/
DROP PROCEDURE [dbo].[produtoAlteraTipo]
GO

/****** Object:  StoredProcedure [dbo].[produtoAlteraTipo]    Script Date: 24-10-2013 12:19:24 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO



create procedure [dbo].[produtoAlteraTipo]
	@cod			[int],
	@tipo			[char](1)
as
begin
	SET XACT_ABORT ON
	BEGIN TRANSACTION T_produtoAlteraTipo
		DECLARE @oldtipo [char](1)

		SELECT @oldtipo = tipo
		FROM [dbo].[produto]
		WHERE cod=@cod

		IF @oldtipo <> @tipo
		BEGIN
			DECLARE @preco [money],
					@qtStock [int],
					@qtMinStock [int]

			UPDATE [dbo].[produto]
			SET	tipo=@tipo
			WHERE cod=@cod

			IF @oldtipo = 'C' AND @tipo = 'D'
			BEGIN

				SELECT TOP 1
					@preco = preco,
					@qtStock = qtStock,
					@qtMinStock = qtMinStock
				FROM ProdutoCrianca
				WHERE cod=@cod

				DELETE FROM ProdutoCrianca
				WHERE cod=@cod

				INSERT INTO ProdutoDesp
```

```
                                        ([cod]
                                        ,[preco]
                                        ,[qtStock]
                                        ,[qtMinStock])
                                VALUES
                                        (@cod
                                        ,@preco
                                        ,@qtStock
                                        ,@qtMinStock)

                        END
                        IF @oldtipo = 'D' AND @tipo = 'C'
                        BEGIN
                                SELECT TOP 1
                                        @preco = preco,
                                        @qtStock = qtStock,
                                        @qtMinStock = qtMinStock
                                FROM ProdutoDesp
                                WHERE cod=@cod

                                DELETE FROM ProdutoDesp
                                WHERE cod=@cod

                                INSERT INTO ProdutoCrianca
                                        ([cod]
                                        ,[preco]
                                        ,[qtStock]
                                        ,[qtMinStock])
                                VALUES
                                        (@cod
                                        ,@preco
                                        ,@qtStock
                                        ,@qtMinStock)

                        END
                END


        COMMIT TRANSACTION T_produtoAlteraTipo
        RETURN
end


GO
```

[NÃO PEDIDO]: Alteração de Tipo de Produto dentro do Trigger de Update da View de Produto

```
CREATE TRIGGER [dbo].trgUpd_ViewProduto
ON [dbo].ViewProduto
INSTEAD OF UPDATE
AS
BEGIN
 BEGIN TRANSACTION
 -- primeiro processa os que mudaram de loja
        -- Eram de crianca
  if exists (select old.codProd
```

```
        from [dbo].Produto old, INSERTED new
       where old.codProd = new.codProd
        and old.tipo <> new.tipo
        and old.tipo = 'c')
 BEGIN
   delete [dbo].ProdutoCrianca
   where
    codProd =
    (Select new.codProd from INSERTED new, [dbo].produto old
       where old.codProd = new.codProd
        and old.tipo <> new.tipo
        and old.tipo ='c');

    -- coloca em Desporto
   insert into [dbo].ProdutoDesporto (codProd, preco, qtStock, qtMinStock)
    select new.codProd, new.preco, new.qtStock, new.qtMinStock
    from INSERTED new, [dbo].Produto old
    where old.codProd = new.codProd
     and new.tipo = 'd'
     and old.tipo = 'c';
 END
    -- Eram de Desporto
if exists (select old.codProd
        from [dbo].Produto old, INSERTED new
       where old.codProd = new.codProd
        and old.tipo <> new.tipo
        and old.tipo = 'd')
 BEGIN
   delete [dbo].ProdutoDesporto
   where
    codProd =
    (Select new.codProd from INSERTED new, [dbo].produto old
       where old.codProd = new.codProd
        and old.tipo <> new.tipo
        and old.tipo ='d');

    -- coloca em Desporto
   insert into [dbo].ProdutoCrianca (codProd, preco, qtStock, qtMinStock)
    select new.codProd, new.preco, new.qtStock, new.qtMinStock
    from INSERTED new, [dbo].Produto old
    where new.codProd = old.codProd
     and new.tipo = 'c'
     and old.tipo = 'd';
 END

--  faz o update dos dados nas tabelas remotas.
-- (mesmo nas que acabou de inserir, mas não deve ter impacto negativo por isso)
UPDATE [dbo].ProdutoCrianca
   Set
      preco = INSERTED.preco,
      qtStock = INSERTED.qtStock,
      qtMinStock = INSERTED.qtMinStock
   FROM INSERTED
   WHERE INSERTED.codProd = produtoCrianca.codProd
    AND INSERTED.tipo = 'c';
```

```
   UPDATE [dbo].ProdutoDesporto
      Set
         preco = INSERTED.preco,
         qtStock = INSERTED.qtStock,
         qtMinStock = INSERTED.qtMinStock
      FROM INSERTED
      WHERE INSERTED.codProd = produtoDesporto.codProd
       AND INSERTED.tipo = 'd';

   UPDATE [dbo].Produto
      set
         codForn =  INSERTED.codForn,
         qtEncomenda=  INSERTED.qtEncomenda,
         estado=  INSERTED.estado,
         tipo =  INSERTED.tipo
      From INSERTED
                  where Produto.codProd = INSERTED.codProd;
          Commit Transaction
END
```

Exercício 2

a) Sabendo que transitoriamente as vendas do centro de vendas de artigos para desportistas também têm de ser realizadas no centro de vendas para crianças, realize as alterações necessárias para que isso seja possível. Discuta o impacto desta alteração sobre o código desenvolvido na alínea b) do ponto 1 e sobre as aplicações que fazem acesso global à base de dados.

b) Sabendo que, por razões de natureza logística, a empresa resolveu juntar, definitivamente, os dois centros de vendas, realize as alterações necessárias ao código desenvolvido no ponto 1 e construa um "script" SQL que permita juntar os dois centros de vendas. Discuta o impacto desta alteração sobre o código desenvolvido na alínea b) do ponto 1 e sobre as aplicações que fazem acesso global à base de dados.

c) Por dificuldades económicas, a empresa resolveu juntar, definitivamente, o centro de vendas com a sede. Realize as alterações necessárias e construa um "script" SQL que permita juntar o centro de vendas com a sede. Discuta o impacto desta alteração sobre o código desenvolvido na alínea b) do ponto 1 e sobre as aplicações que fazem acesso global à base de dados.

Resolução

Alinea a)

No contexto definido para o Excercicio 1, os *updates* nos Centros de Vendas são feitos directamente sobre as respectivas tabelas de Produtos (existentes em cada loja) enquanto que os *updates* realizados na instancia da Sede são feitos sobre a *view* `viewProduto (com uso do respectivo trigger de instead of update)` a qual foi construída com o intuito de centralizar toda a informação de produtos.

A venda de produtos de Desportita no centro de vendas de criança, obriga a que este tenha acesso também à tabela com produtos de Desportita. Mantendo a permissa de que as lojas podem fazer vendas (operar) mesmo na ausência de ligações com a sede, então uma solução é a implementação de um mecanismo de replicação entre os centros de vendas, por forma a que para além de o centro de vendas de crianças poder efectuar vendas sobre produtos de desporto, esta venda é replicada para o centro de vendas de desporto.

Esta implementação não terá impacto funcional sobre o desenvolvimento do exercicio 1 b) nem sobre o acesso global aos dados, contudo como a replicação é feita asincronamente entre os dois centros de venda, poderá existir um desfasamento pouco significativo entre os 2 centros de vendas.

P2P Transaction Replication

INST2 -> Centro Vendas Criancas
INST3 -> Centro Vendas Desporto

1) Config Local Distribution @ INST2
2) Config Local Distribution @ INST3
3) Config Publication P2P @ INST2
4) Config P2P Topology Add INST3 @ INST2

Não ter impacto funcional sobre o desenvolvimento significa que não será necessário alterar o código anterior que usava as vistas e os procedimentos que tinham sido disponibilizados (ou seja existe uma grande independencia entre operações e código/ desenvolvimento).

Alinea b)

A forma mais linear de excutar esta alteração era na Loja Conjunto criar uma tabela com os produtos das anteriores lojas, criando uma tabela adicional que ligaria CodProd a Tipo. Esta tabela permitiria substituir os anteriores sinónimos das tabelas de Produtos usados na Sede por vistas a correr sobre a mesma tabela ProdutosLoja.

A tabela de Tipos na Loja seria manipulada por triggers a incluir na nova view.

Provávelmente esta seria a forma de fazer menos código, teria a vantagem de ser mais facilmente reversível, mas a grande desvantagem do desempenho (e complexidade do código).

A melhor solução seria, juntar os produtos numa nova tabela para servir a Loja a ProdutosLoja, que incluiria além dos campos actuais também o tipo, e alterar as views e procedimentos em uso na Sede para eliminar os "if" usados na seleção de loja e os updates para cada loja.

O campo Tipo, poderia ser retirado da tabela de produtos da Sede e ficaria apenas na Loja (agora já não vantagem em estar na Sede e do ponto de vista logico e operacional faz falta na Loja).

Mantendo as vistas e os procedimentos, não seria necessário alterar o código que estivesse a usar a base de dados.

Alinea c)

A solução seria uma evolução do ponto anterior:

- Acrescentar na tabela de Produtos da sede as restantes colunas
- Correr um update para as actualizar com base na informação da tabela da Loja
- Substituir a ViewProduto por um sinónimo eliminando o trigger de upgrade que deixa de ser necessário
- Rever o codigo dos procedimentos para fazerem todas as manipulações numa única tabela

Ficariamos com o modelo lógico global definido no inicio do excercicio 1, mas com os procedimento e vista ViewProdutos mantidos para evitar alteração do código que aceda à base de dados.

Exercício 3

Considere novamente a caracterização inicial do problema. Pretende-se acrescentar ao modelo lógico global o registo de ocorrências associadas a produtos, sendo cada ocorrência caracterizada por um identificador (identity) e um texto descritivo, estando relacionada com o produto a que se refere. Pretende-se que em cada centro de vendas se possa, de forma autónoma, consultar e manipular ocorrências relativas aos produtos nele contidos. Igualmente, pretende-se que na sede se possam realizar consultas e manipulações das ocorrências dos produtos (dos dois tipos). Admitem-se discrepâncias entre os dados das ocorrências na sede e em cada centro de vendas, desde que, em situações normais, elas sejam ultrapassadas em alguns segundos.
Implemente uma solução que permita cumprir estes objectivos.

Resolução

Ensaio com Replicação Peer-to-Peer

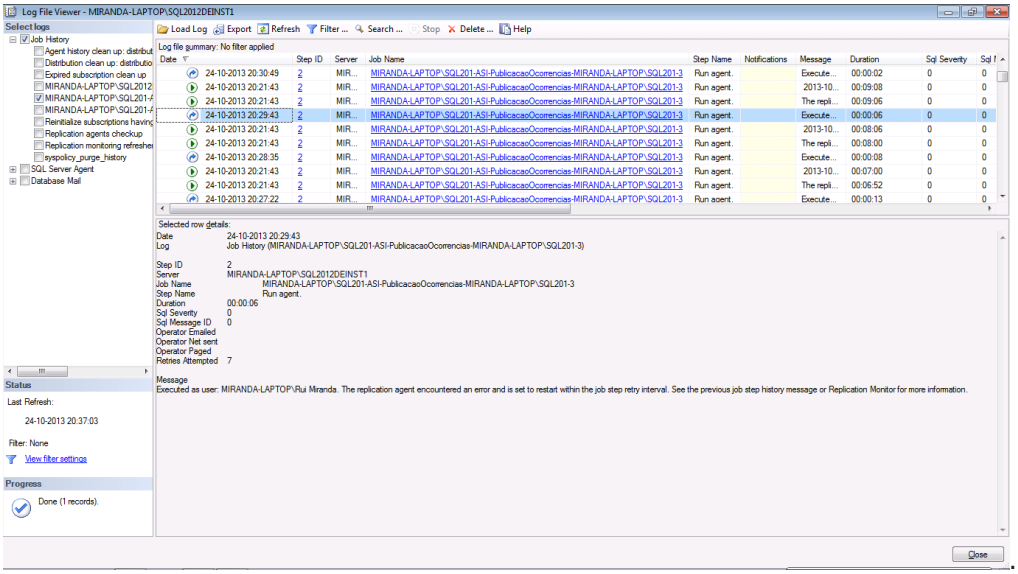
Para resolver o indicado neste exercício optámos pela replicação Peer-to-Peer criando uma estrutura em estrela entre a tabela criada na sede e cada uma das Lojas / centro de vendas. Tivemos o cuidado de usar na "Identidade" campo do tipo "identity", uma semente diferente para cada Instância da tabela Ocorrências e um incremento igual a 3 para todos (foi o n. máximo de nós que consideramos, mas poderia ser um n. superior caso pretendêssemos incluir mais nós).

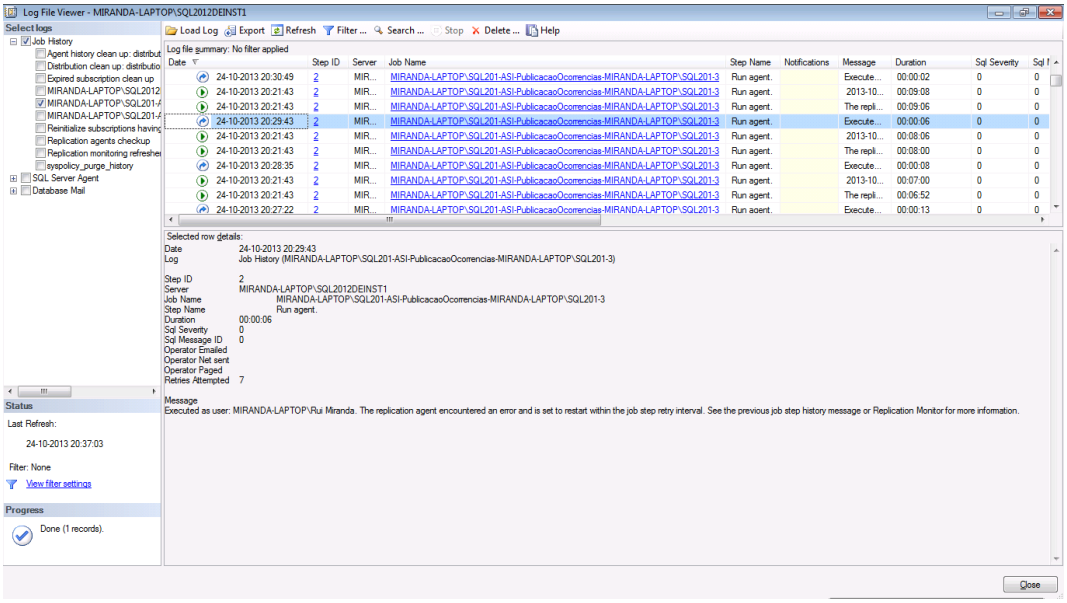
Depois de montar o ambiente, verificamos que **não funcionava** e o porquê. A replicação Peer-to-Peer tem algumas limitações que estão documentadas pela Microsoft (http://technet.microsoft.com/en-us/library/ms151196.aspx):

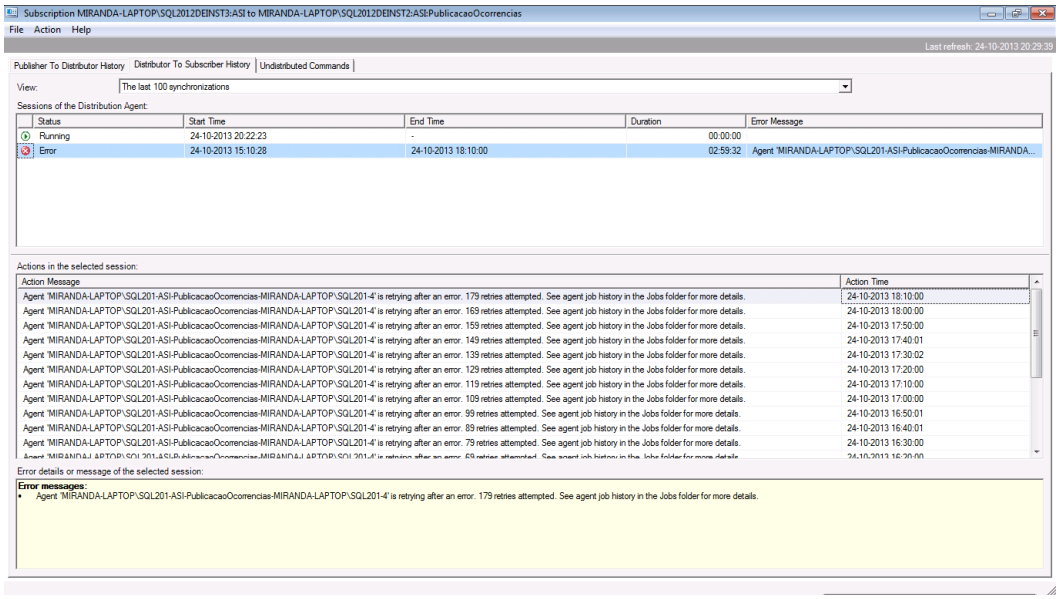
- Initialization and reinitialization with a snapshot.
- Row and column filters.
- Timestamp columns.
- Non-SQL Server Publishers and Subscribers.
- Immediate updating and queued updating subscriptions.
- Anonymous subscriptions.
- Partial subscriptions.
- Attachable subscriptions and transformable subscriptions. (Both of these options were deprecated in SQL Server 2005.)
- Shared Distribution Agents.
- The Distribution Agent parameter **-SubscriptionStreams** and the Log Reader Agent parameter **-MaxCmdsInTran**.
- The article properties **@destination_owner** and **@destination_table**.

O erro obtido na replicação foram

> Agent 'MIRANDA-LAPTOP\SQL201-ASI-PublicacaoOcorrencias-MIRANDA-LAPTOP\SQL201-3' is retrying after an error. 7 retries attempted. See agent job history in the Jobs folder for more details.
> Executed as user: MIRANDA-LAPTOP\Rui Miranda. The replication agent encountered an error and is set to restart within the job step retry interval. See the previous job step history message or Replication Monitor for more information.







e isto resulta do facto do nó SEDE tentar replicar para uma loja um Produto que não existe na tabela local de produtos (que só tem os usados na loja).

Ensaio com Replicação Peer-to-Peer c/ Vistas com Filtro Horizontal

O ensaio seguinte foi o de tentar filtrar na origem (na tabela de Ocorrências na Sede) as linhas a enviar para cada Loja, em função do tipo de artigo. Construir uma view com filtro sobre a coluna tipo (uma para cada tipo de artigo) e que permitisse inserir registos foi relativamente fácil, mas neste caso não foi sequer possível montar a Topologia de replicação Peer-to-Peer. A razão esta na lista acima "filtro por linha".

Outras soluções alternativas

Outras soluções possíveis são a replicação integral em cada um dos nós Loja da tabela de Produtos, ou o uso de "Updatable Subscription for Transactional Replication" descrita em http://technet.microsoft.com/en-us/library/ms151718.aspx, mas que não chegamos a conseguir ensaiar, a tempo para incluir no relatório.

Apenas para documentar o ensaio incluímos no relatório o que fizemos para montar esta replicação.

Opção Criação de replicação Peer-to-Peer

1. Criar as tabelas de Ocorrencias nos diferentes nós (manualmente para poder controlar a semente e o salto no campo identity, que é a chave única)
2. na base de dados da SEDE, adicionar uma "Replication/Local Publications"
3. parametrizar a instancia para ser o seu próprio distribuidor
4. configurar arranque do agente e snapshot folder
5. para o tipo de publicação seleccionar "Peer-to-Peer publication"
6. no ecran de configuração da Topologia Peer-to-Peer incluir os nós
7. ligar os nós para os quais se quer replicação
8. gerar os script ou executar logo a configuração

Apresentamos a seguir os scripts usados

SEDE: Criação da Tabela de Ocorrências

```
--03.0-CREATE-SEDE-TBL-ocorrencias.sql
USE [ASI]
GO
ALTER TABLE [dbo].[ocorrencias] DROP CONSTRAINT [FK__ocorrenci__codPr]
GO
DROP TABLE [dbo].[ocorrencias]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[ocorrencias](
        [identificador] [int] IDENTITY(0,3) NOT FOR REPLICATION NOT NULL,
        [descricao] [text] NULL,
        [codProd] [int] NULL,
PRIMARY KEY CLUSTERED
(
        [identificador] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
ALTER TABLE [dbo].[ocorrencias]  WITH CHECK ADD  CONSTRAINT [FK__ocorrenci__codPr] FOREIGN KEY([codProd])
REFERENCES [dbo].[produto] ([cod])
```

```
GO
ALTER TABLE [dbo].[ocorrencias] CHECK CONSTRAINT [FK__ocorrenci__codPr]
GO
```

Centro de Venda de Produtos de Criança: Criação da Tabela de Ocorrências

```
--03.1-CREATE-CVCrianca-TBL-ocorrencias.sql
USE [ASI]
GO
ALTER TABLE [dbo].[ocorrencias] DROP CONSTRAINT [FK__ocorrenci__codPr]
GO
DROP TABLE [dbo].[ocorrencias]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[ocorrencias](
	[identificador] [int] IDENTITY(1,3) NOT FOR REPLICATION NOT NULL,
	[descricao] [text] NULL,
	[codProd] [int] NULL,
PRIMARY KEY CLUSTERED
(
	[identificador] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
ALTER TABLE [dbo].[ocorrencias]  WITH CHECK ADD  CONSTRAINT [FK__ocorrenci__codPr] FOREIGN KEY([codProd])
REFERENCES [dbo].[produto] ([cod])
GO
ALTER TABLE [dbo].[ocorrencias] CHECK CONSTRAINT [FK__ocorrenci__codPr]
GO
```

Centro de Venda de Produtos de Desporto: Criação da Tabela de Ocorrências

```
--03.2-CREATE-CVDesp-TBL-ocorrencias.sql
USE [ASI]
GO
ALTER TABLE [dbo].[ocorrencias] DROP CONSTRAINT [FK__ocorrenci__codPr]
GO
DROP TABLE [dbo].[ocorrencias]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[ocorrencias](
	[identificador] [int] IDENTITY(0,3) NOT FOR REPLICATION NOT NULL,
	[descricao] [text] NULL,
	[codProd] [int] NULL,
PRIMARY KEY CLUSTERED
(
	[identificador] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
ALTER TABLE [dbo].[ocorrencias]  WITH CHECK ADD  CONSTRAINT [FK__ocorrenci__codPr] FOREIGN KEY([codProd])
```

```
REFERENCES [dbo].[produto] ([cod])
GO
ALTER TABLE [dbo].[ocorrencias] CHECK CONSTRAINT [FK__ocorrenci__codPr]
GO
```

FIM