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

Mestrado em Engenharia Informática e de Computadores
Arquitetura de Sistemas Distribuídos
2013/2014
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

Introdução

Exercício 1

a) Conceba os esquema lógico global e de fragmentação e distribua a base de dados por três instâncias do Sql Server, de acordo com o esquema de fragmentação adotado.

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 atualizações de produtos devem poder ser realizadas através da execução direta 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.

d) Construa o código que, na sede, permita receber uma encomenda de uma dada quantidade de um produto.

e) Construa o código que, na sede, permita alterar o tipo de um produto.

Alínea a)

Objetivo

O objetivo do esquema Logico Global é apresentar a base de dados distribuída como se de uma única instância se tratasse.

Os esquemas de Fragmentação e de Distribuição, são visões mais detalhadas, que levam em conta a fragmentação dos dados impostos pelos requisitos do ambiente distribuído, nomeadamente requisitos relativos ao funcionamento autónomo dos nós suportados nas instâncias distribuídas.

Implementação

Na implementação do fracionamento Horizontal foi determinada pela necessidade de cada loja operar de forma autónoma (mesmo sem ligação ativa à Sede), enquanto a fragmentação vertical foi determinada pelas operações que precisavam de ser feitas nas lojas ou na sede e os atributos necessários ao seu suporte. Tendo em conta estas necessidades e a procura de um bom desempenho, em termos de implementação optámos por:

- Colocar o "tipo" na tabela da sede, porque permite escolher a que loja pertence determinado produto e com base nisso envolver apenas a instância correspondente numa operação de alteração ou delete de registo;
- Colocar os campos "qtStock" e "qtMinStock" na mesma tabela, para permitir a filtragem remota (na instância da loja) os registos e considerar para encomenda. Caso os campos estivessem em instâncias separadas a totalidade da tabela teria de ser enviada para a sede, pois localmente não haveria forma de executar qualquer filtro.

Esquema Logico Global

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

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)

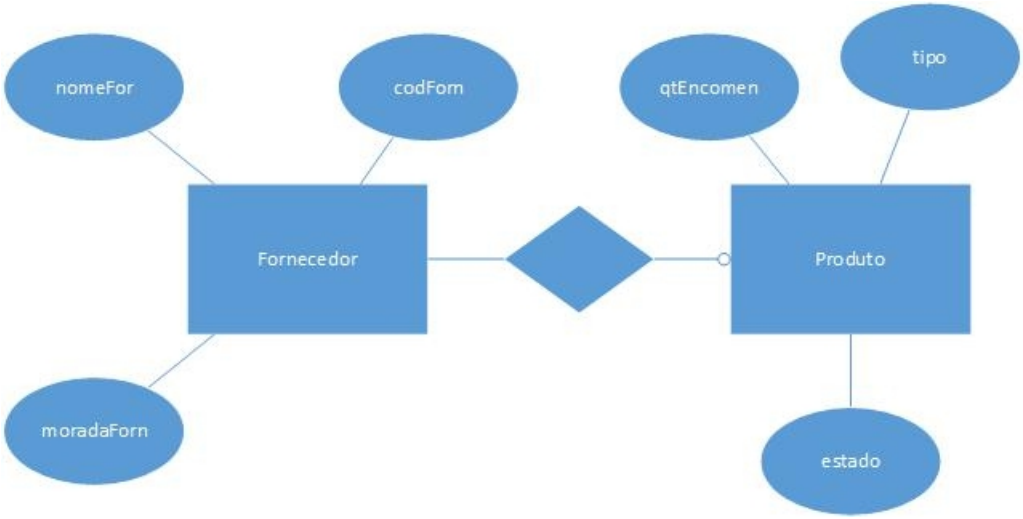
Partição Horizontal (centros de vendas)

ProdutoLojas(codProd, tipo, preço, qtStock)

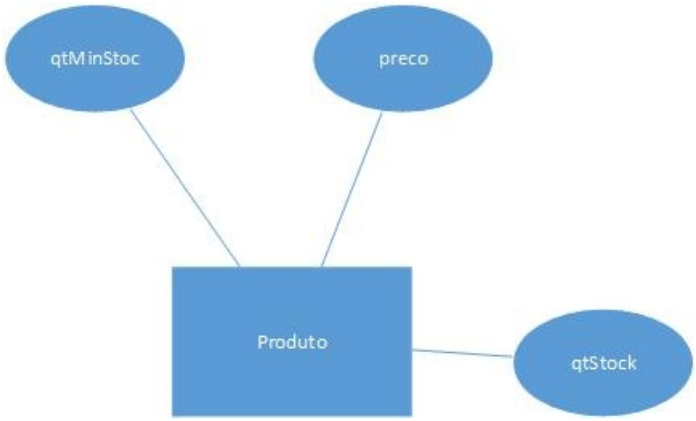
- ProdutoLojaDesp = σ{tipo = 'desporto'}(ProdutoLojas)
- ProdutoLojaCriança = σ{tipo = 'criança'}(ProdutoLojas)

Em termos do modelo ER, ficamos os seguintes diagramas:

SEDE – modelo ER



Loja – modelo ER



Na implementação foi usado o ficheiro BAT "01.a.run.bat" que permite criar todos os objetos necessários à solução da alínea a) do exercício 1.

Este script usa o ficheiro "..\00.config_inst.txt" onde são indicadas as instâncias de cada uma das bases de dados envolvidas na distribuição:

- SEDE – Indica a instância onde é feita a simulação da sede;

- CVCr – Indica a instância do Centro de Venda de Vestuário de Criança, que muitas vezes referimos como Loja de Crianças;
- CVDp – Indica a instância do Centro de Venda de Vestuário de Desportista, que muitas vezes referimos como Loja de Desportistas.

O script "01.a.run.bat" usa o comando SQLCMD para executar os diversos scripts sql, que são indicados abaixo e cujo nome indica (mais ou menos) o que vão fazer. A implementação termina com um teste para validar os elementos criados. Os scripts sql usados são:

- 01.a.2-CREATE-SEDE.sql
- 01.a.3-CREATE-LOJA.sql – Este script é chamado 2 vezes uma para cada loja
- 01.a.4-CREATE-SEDE-LINKEDSERVERS.sql
- 01.a.5-TEST.sql

No processo de criação, criamos 12 produtos, 6 de Criança e 6 de Desportista e 3 fornecedores.

As instâncias ficaram com as seguintes objetos:

SEDE	*CVVC (loja crianças)*	*CVVD (loja desportistas)*
Tabelas: Fornecedor Produto Sinónimos: ProdutoCrianca ProdutoDesp	Tabelas: produto	Tabelas: produto

Conclusões

O esquema Lógico Global da Base de Dados, apresenta uma visão global da base de dados, como se fosse uma única e instância, com transparência à fragmentação e desta à distribuição dos dados. Os processamentos centrais (na sede) estão suportados na implementação do esquema Lógico Global e por isso não precisam de ter em conta a fragmentação e a distribuição dos dados. Esses pontos (conhecimento da fragmentação e da distribuição) são apenas tratados dentro das Vistas e Procedimentos usados na implementação do esquema Lógico Global.

O esquema de Distribuição leva em conta as necessidades operativas de cada loja (autonomia) e ao esquema de Fragmentação a localização das operações e os dados de que necessitam. Um elemento importante é a considerar é também o desempenho da totalidade do sistema e é possível adequar a localização ou mesmo replicar alguns atributos por razões e desempenho. Uma das razões pode mesmo ser o permitir filtragens locais e minimizar os registos a passar entre os nós nas operações globais.

A utilização de linkedservers e sinónimos permite ter transparência à localização dos dados distribuídos.

Alínea b)

Objetivo

O objetivo é o desenvolvimento de uma solução que permita acesso global com base no esquema lógico global com níveis adequados de independência face aos esquemas de fragmentação e de distribuição.

Usando procedimentos para a inserção e remoção de produtos, uma vista agregadora para as operações de consulta, com um gatilho associado para as operações de update.

Implementação

A implementação incluiu uma vista ViewProduto que permite ver todos os produtos e seus atributos de forma transparente à sua fragmentação e distribuição. Para utilização da vista nas instruções de "update" foi criado um "trigger instead of update" sobre a vista, que tomava em conta a fragmentação e a distribuição da tabela produtos, embora usando os sinónimos para ter independência face à localização física das tabelas fora da sede.

Os procedimentos foram implementados tendo em conta, tal como na vista, a fragmentação e a distribuição lógica dos dados usando também os sinónimos criados (independência da localização física).

Na implementação desta alínea foi usado um o comando "01.b.run.bat" que executa, com o sqlcmd os seguintes scripts:

- 01.b.0-CREATE-SEDE-SP-insereProduto-removeProduto.sql
- 01.b.1-CREATE-SEDE-VW-viewProduto.sql
- 01.b.2-CREATE-SEDE-TRG-trViewProduto.sql
- 01.b.3-TEST.sql
- 01.b.4-TEST-viewProduto.sql

resultando na adição dos seguintes elementos à instância SEDE:

SEDE	*CVVC (loja crianças)*	*CVVD (loja desportistas)*
Vistas: viewProduto trViewProduto Procedimentos insereProduto removeProduto		

No teste aos novos elementos foram inseridos 2 novos produtos usando o procedimento criado.

A vista foi usada em update, na simulação de uma venda de 40% do stock de produtos existentes.

Conclusões

Os elementos criados na alínea, contribuem para o Esquema Lógico Global com a transparência de Fragmentação e de Distribuição.

A vista criada apresenta uma visão global dos produtos com completa transparência à sua fragmentação e distribuição pelas diferentes instâncias. O mesmo se aplica ao procedimento de inserção a usar na introdução de novos produtos.

A vista foi dotada de um gatilho (trigger de instead of update) que permite o seu uso para as operações de update (expecto do campo de tipo de produto) com total transparência, permitindo usar a vista como uma tabela, com total transparência à fragmentação e distribuição.

O gatilho de update poderia ter também implementada a operação de update ao tipo de produto e a vista poderia também ter gatilhos para as operações de insert e delete.

Estas operações podem igualmente ser implementadas com procedimentos, como foi a opção para o presente enunciado.

Alínea c) e d)

Objetivo

Criar 2 procedimentos que, na sede, permitam colocar e receber encomendas. Os procedimentos precisam de simultaneamente usar e alterar dados da sede e das lojas, fazendo-o sincronamente dentro de uma transação distribuída.

Sendo a operação global executada na Sede poderá ser usado o esquema Lógico Global para a implementação dos procedimentos.

Implementação

Este conjunto de alíneas foi implementado com base no script bat: 01.dc.run.bt , que chamam o c comando sqlcmd para executar os seguintes scripts sql:

- 01.c.0-CREATE-SEDE-ENCOMENDAS.sql
- 01.d.0-CREATE-SEDE-SP-produtoEncomendadoFoiRecebido.sql
- 01.dc.1-TEST.sql

Conclusões

A utilização da vista de produtos e do seu gatilho de update (esquema lógico global da base de dados) permite implementar os procedimentos aproveitando a transparência de fragmentação e distribuição proporcionadas.

Alínea e)

Objetivo

O procedimento a criar nesta alínea tinha como foco o mover um produto entre 2 lojas (de uma instância para a outra) controlado a partir da Sede. Esta transação sincrona envolve as 3 instâncias e utiliza os LinkedServer criados.

Implementação

Na implementação usámos os Linkedserver para escrever nos fragmentos horizontais, mas decidimos escrever diretamente na tabela de produtos da Sede, por uma questão de desempenho (evitando a vista). Uma vez que se pretendeu ter um procedimento independente da vista para a alteração do tipo de produto, parecia um contrassenso usar a vista para apenas fazer update desse campo na tabela da sede.

Esta opção com foco no desempenho, tem um preço a pagar que é a perca de transparência relativamente à fragmentação (vertical).

Esta alínea foi implementada com base no script bat, 01.e.run.bat, que chama o comando sqlcmd para executar os seguintes scripts sql:

- 01.e.0-CREATE-SEDE-SP-produtoAlteraTipo.sql

- 01.e.1-TEST.sql

Nos testes foi testada a mudança de diversos produtos entre as 2 lojas.

Conclusões

O procedimento implementado usa os LinkedServers criados aproveitando a sua transparência à Distribuição (transparência à localização dos fragmentos horizontais), mas porque usa diretamente a tabela de produto da sede (para atualizar o tipo) não tem essa transparência em relação à fragmentação vertical.

Caso tivesse sido implementada a mudança de tipo também no gatilho de update da vista, em caso de alteração da fragmentação vertical seria apenas necessário alterar a vista, enquanto neste caso será necessário alterar também este procedimento.

Exercício 2

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.

Contexto

O contexto ao início do exercício era caracterizado de acordo com a tabela abaixo:

SEDE	*CVVC (loja crianças)*	*CVVD (loja desportistas)*
Tabelas: fornecedor produto produtosEncomendados Sinónimos: ProdutoCrianca ProdutoDesp Vistas: Produto Gatilho: trgViewProduto Procedimentos insereProduto produtoAlteraTipo encomendarProdutos	Tabelas: produto	Tabelas: produto

SEDE	*CVVC (loja crianças)*	*CVVD (loja desportistas)*
receberProduto removeProduto		

Alínea a)

Objetivo

Esta alínea do exercício tinha por objetivo a migração temporária das vendas da loja de desportistas (CVVD) para a loja de crianças (CVVC).

O ponto determinante aqui é o ser temporário, o que implica a escolha de uma solução que seja também facilmente reversível.

Não temos informação sobre os programas de vendas usados nas lojas, mas parece razoável assumir que, com base num código de barras (por exemplo) obtêm a identificação do produto, que permite aceder à tabela, obter ou confirmar o preço e atualizar o stock. Nesse caso era importante continuar a fornecer ao programa um "objeto" da base de dados que tivesse agora todos os produtos à venda na loja. Esse novo "objeto" seria uma view em substituição da tabela de produtos usada até então.

Implementação

Esta implementação implicava uma indisponibilidade temporária da venda de produtos de vestuário para Desportistas e de Crianças passou por:

- criar na base de dados do CVCr uma tabela com os produtos para desportistas que estava em CVDp;
- criar uma nova Vista para juntar as 2 tabelas numa visão única para os programas das lojas;
- alterar os sinónimos na sede para apontarem para as novas tabelas e localizações.

O código que implementamos para este exercício é ativado a partir de um ficheiro de batch que evoca cada uma das alterações realizadas e é composto pelos seguintes ficheiros:

- 02.a.run.sql -
- 02.a.0-CREATE-SEDE-Desp Temp-At-CVCr.sql -
- 02.a.1-Copy-Sede-ProdutoDesp-CVCr.ProdutoDespTemp.sql
- 02.a.2-CREATE-Sede-Synonym
- 02.a.3-CREATE-Sede-Produto-View-Trig
- 02.a.4-TEST-CVCr
- 02.a.5-SEDE-Revert

Conclusões

A solução implementada envolve apenas atividade de administração de base de dados (DBA), não necessitando de alterações aos programas usados na sede e na loja,

As alterações realizadas tiveram impacto apenas no "Esquema de Distribuição" e o uso de sinónimos para as tabelas remotas permitiu ter independência face à sua localização.

A vista para a visão global dos produtos e os procedimentos de inserção e delete de um produto, porque usam os sinónimos criados não precisam de ser alteradas, nem tão pouco os programas que os usem.

Alínea b)

Objetivo

O objetivo desta atividade é a junção de forma definitiva das duas lojas num único centro de venda. Sendo uma junção "definitiva", podemos proceder a algumas alterações tendo por foco simplificar a manutenção e a melhoria da performance do sistema total.

Assim esta alteração passará, por na nova loja, juntar todos os produtos numa única tabela, alterar a vista e os procedimentos implementados no ponto 1.b) para passarem a usar uma única tabela (via sinónimo ProdutoCrianca que se vai manter). O sinónimo "produtoDesp" deixa de ser necessário e será eliminado.

Implementação

Para a implementação desta alínea usámos o ficheiro "02.b.run.bat", que ativa via sqlcmd os seguintes scripts sql:

- 02.b.1-CREATE-SEDE-Sinonimo-FusaoPartHoriz.sql
- 02.b.2-CREATE-SEDE-VW-viewProduto.sql
- 02.b.3-CREATE-SEDE-TRG-trViewProduto.sql
- 02.b.4-CREATE-SEDE-SPs-insereProduto-removeProduto.sql
- 02.b.5-CREATE-SEDE-SP-produtoAlteraTipo.sql
- 02.b.5-TEST-Sede.sql
- 02.b.7-TEST-CVCr.sql

Por uma questão de coerência com os nomes optámos por eliminar também o sinónimo de ProdutoCrianca e criar um denominado por "ProdutoCV" para onde antes apontava o sinónimo criança.

Conclusões

A solução implementada apesar de fazer grandes alterações na vista, no gatilho e nos procedimentos de criação, não implica qualquer alteração para os programas, que os usam na sede. Temos por transparência à fragmentação e à distribuição.

Alínea c)

Objetivo

O objetivo desta atividade é a junção de forma definitiva da base de dados distribuída numa única instância de base dado na sede.

A tabela de produtos passará a ser única servindo as funções necessárias na sede e na loja.

A vista sobre os produtos poderá ser substituída por um sinónimo e o gatilho apagado.

A tabela em uso nas lojas poderá ser substituída por uma vista contendo apenas os atributos necessários aos programas que operam na componente loja.

Por questões de compatibilidade os procedimentos de criação, remoção e alteração de tipo podem ser mantidos, mas simplificados para usar apenas uma tabela de produtos.

Implementação

Para a implementação desta alínea usámos o ficheiro "02.c.run.bat", que ativa via sqlcmd os seguintes scripts sql:

- 02.c.1-Fusao-produto-SEDE-CV.sql
- 02.c.2-TEST-Fusao-produto-SEDE-CV.sql

Conclusões

A solução implementada, permite às anteriores aplicações da sede continuarem a funcionar sem precisarem de qualquer alteração, sendo que agora o esquema lógico global coincide com o esquema lógico da própria instância. Sendo uma alteração definitiva, a junção na sede foi conveniente juntar os dados e rever procedimentos e vista para melhorar o desempenho (e simplificar a administração e manutenção da instância).

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.

Objetivo

Neste último exercício decidimos que a melhor solução consistia em implementar uma Replicação Peer-to-Peer de forma a garantir que o registo de ocorrências é propagado pelas várias instâncias de base de dados, garantindo assim que as várias instâncias de base de dados contêm todas as ocorrências independentemente do local onde estas são inseridas.

Implementação

Para a implementação desta alínea usámos o ficheiro "3.1. Setup tabelas de Ocorrencia e Distribuidores.bat", que ativa via sqlcmd os seguintes scripts sql:

- 3.1.1. Setup tabelas de Ocorrencia @SEDE.sql
- 3.1.2. Setup tabelas de Ocorrencia @CVDp.sql
- 3.1.3. Setup tabelas de Ocorrencia @CVCr.sql
- 3.1.4. Configurar Distribuidores.sql

Cada um dos primeiros três *scripts*, cria a tabela `ocorrencia`, em cada uma das três instâncias. O motivo que leva à utilização de três *scripts* diferentes prende-se com a definição da coluna *identity*:

```
id integer identity (1, 3) not for replication primary key, -- na SEDE
id integer identity (2, 3) not for replication primary key, -- no CV Criancas
id integer identity (3, 3) not for replication primary key, -- no CV Desporto
```

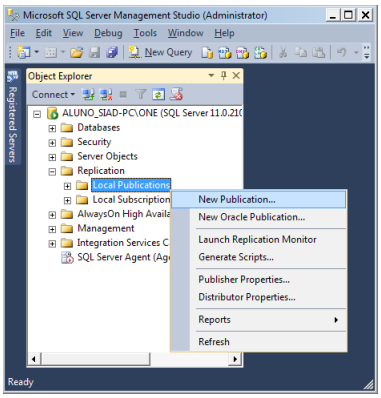
Note-se que o valor de *seed* varia em função da instância. Isto faz-se porque qualquer linha inserida na tabela `ocorrencia` numa dada instância de BD será propagada para as outras duas instâncias. Se não houver diferença no valor de *seed* ocorrem erros durante o processo de

replicação, uma vez que esta coluna é chave primária. Note-se também que o valor de incremento é igual ao número de instâncias em uso. Isto implica que se alguma vez se adicionar uma quarta instância, a coluna `id` tenha que ser reconstruida. Para evitar este problema pode-se simplesmente definir o incremento como sendo o número máximo de instâncias que se prevê utilizar no futuro.

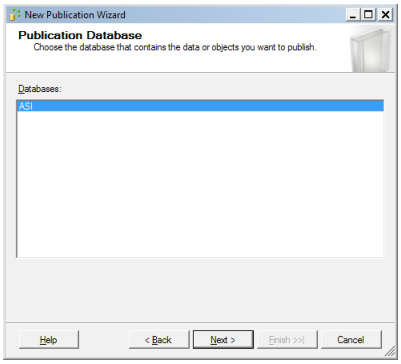
A inclusão da keyword `not for replication` garante que a coluna `id` não é incrementada quando a inserção é realizada por um agente de replicação; apenas as inserções na própria instância irão incrementar esta coluna `identity`.

O quarto *script* configura as três instâncias como distribuidores, em preparação para a criação de publicações e subscrições.

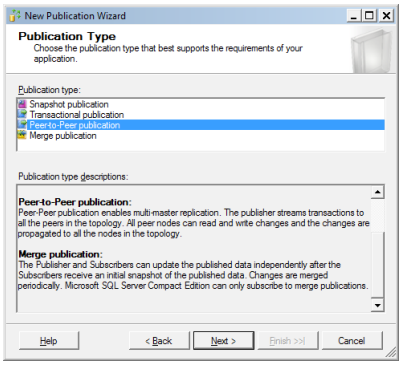
O trabalho de criação de publicações e subscrições infelizmente não pode ser executado via *script* devido à falta de tempo para investigar a utilização dos *stored procedures* que permitir a utilização desta funcionalidade. Assim esta parte do trabalho foi executado com recurso à *GUI* do *MS Management Studio*:



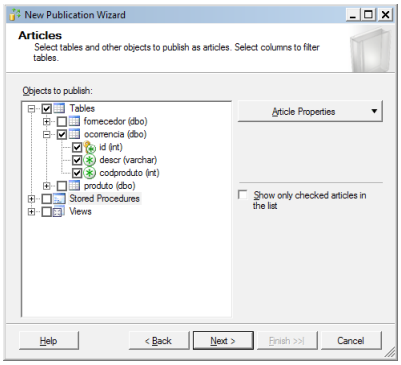
1. Iniciar o *Wizard* de Replicação P2P



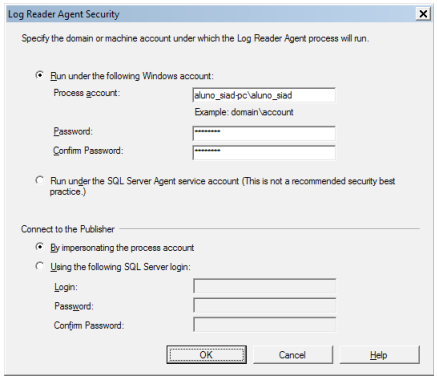
2. Indicar a BD com os artigos a publicar



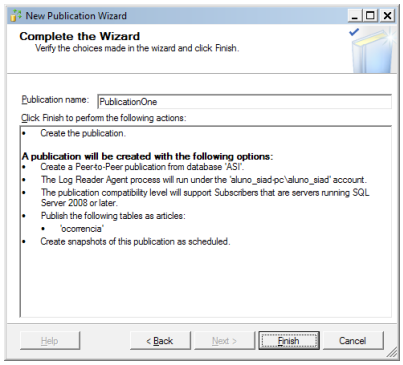
3. Determinar o tipo de publicação



4. Determinar a tabela a publicar

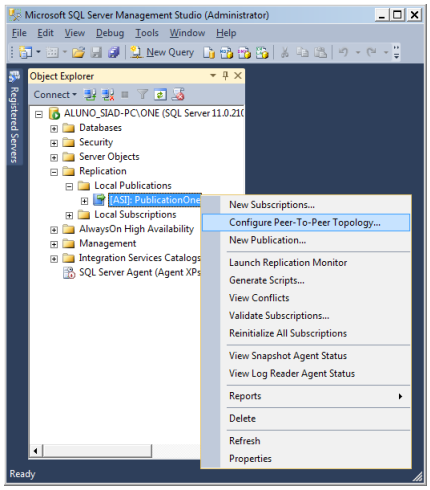


5. Definir credenciais de segurança para a publicação

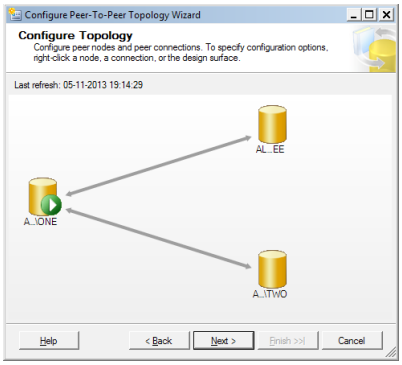


6. Determinar o nome da publicação

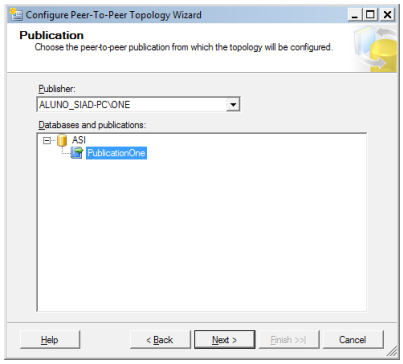
Uma vez configurada a publicação sobre a sede, pode-se configurar as subscrições e as restantes publicações com o auxilio do *Wizard* de configuração da topologia P2P:



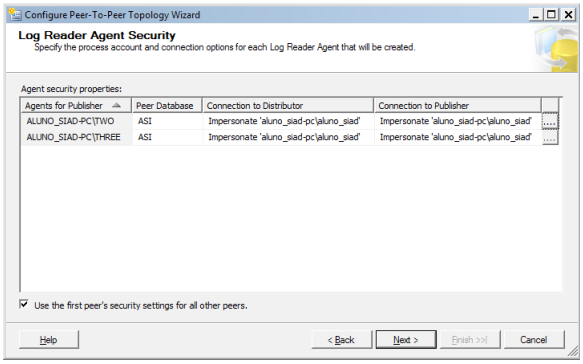
1. Iniciar o *Wizard* para configurar a topologia P2P



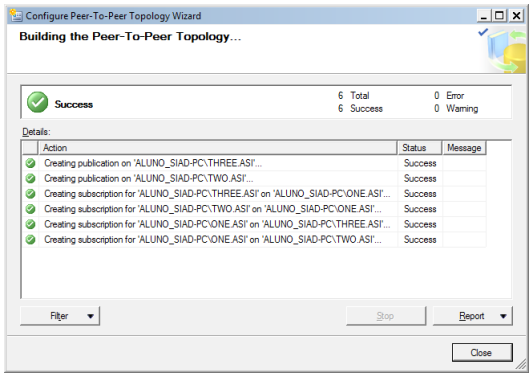
3. Determinar o *layout* da topologia P2P



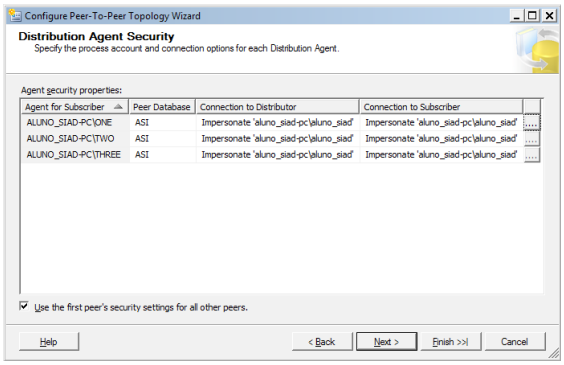
2. Determinar a publicação a distribuir por P2P



4. Definir credenciais de segurança para os *log Readers*



5. Definir credenciais de segurança para os agentes de Distribuição



6. Confirmação de sucesso da construção da topologia

Uma vez realizada esta configuração as instâncias estão prontas para se realizar um teste.

O teste foi implementado sob a forma do ficheiro “3.4. Executar Teste a Replicacao P2P.bat”, que usa o sqlcmd para executar os seguintes scripts sql:

- 3.4.1. Listar Produtos disponiveis.sql
- 3.4.2. Inserir 3 Ocorrencias na Sede.sql
- 3.4.3. Inserir 3 Ocorrencias na CVCr.sql
- 3.4.4. Inserir 3 Ocorrencias na CVDp.sql

Depois de inserir as ocorrências, o *batch* limita-se a contar duas vezes o número de ocorrências existentes em cada uma das instâncias. Uma imediatamente a seguir aos *inserts* e uma segunda vez, alguns segundos depois, para constatar que o processo de replicação já ocorreu.

Conclusões

O processo de replicação permite-nos manter a informação existente numa dada tabela sincronizada entre várias instâncias de bases de dados. Dependendo da forma como a tabela se insere no esquema da base de dados local é necessário ter atenção a alguns detalhes sem os quais o processo de replicação falha durante a sua execução. Exemplos desses cuidados são por exemplo a atenção dada às necessidade da coluna `id` explicada anteriormente ou por exemplo a não aplicação de chaves estrangeiras à tabela de ocorrências nos centros de vendas. Apesar do processo de replicação exigir este nível de atenção, é certamente preferível à alternativa que consistiria em manter os dados sincronizados manualmente, um processo que implicaria muito mais esforço.