

UNIVERSIDADE ESTADUAL DE MARINGÁ
ESPECIALIZAÇÃO EM DESENVOLVIMENTO DE SISTEMA PARA A
WEB

HENRIQUE BUOSI MONTEIRO

APOIANDO UM SISTEMA ADMINISTRATIVO DE UMA
UNIVERSIDADE MULTICAMPI COM O MICROSOFT SQL SERVER
2000

MARINGÁ – PR , 30 DE JUNHO DE 2005

HENRIQUE BUOSI MONTEIRO

ORIENTADOR: ADEMIR CARNIEL

**APOIANDO UM SISTEMA ADMINISTRATIVO DE UMA
UNIVERSIDADE MULTICAMPI COM O MICROSOFT SQL SERVER**

2000

Monografia submetida à Universidade Estadual
de Maringá como parte dos requisitos para
obtenção do grau de Especialista em
Desenvolvimento de Sistema para Web

MARINGÁ – PR , 30 DE JUNHO DE 2005

Dedico esta monografia àqueles que de forma simples e honesta contribuíram para sua realização. Em especial a meus parentes que sempre me incentivou a buscar novos conhecimentos.

Agradecimentos

- Ao Professor Ademir Carniel, orientador, que me acolheu e me conduziu ao longo desta jornada.
- A meus pais, irmãos e minha noiva que souberam compreender a minha ausência nos momentos em que me dedicava a esta tarefa.

SUMÁRIO

Lista de Figuras	viii
Lista de Tabelas	ix
Lista de Abreviaturas.....	x
Resumo	xi
Abstract.....	xi
CAPÍTULO I	12
1.1 Introdução	12
1.2 Objetivo Geral.....	14
1.3 Objetivo Específico.....	14
1.4 Motivação.....	15
1.5 Justificativa	17
1.6 Apresentação do Trabalho Desenvolvido	18
CAPÍTULO II.....	19
2.1 Bancos de Dados	19
2.1.1 Chave Primária (Primary Key).....	21
2.1.2 Chave Estrangeira (Foreign Key).....	22
2.2 Sistemas de Gerenciamento de Bancos de Dados.....	22
2.3 Importância dos SGBD's nos Sistemas Administrativos.....	22
2.4 Fundamentos dos Sistemas Administrativos.....	23
2.5 Sistemas Administrativos com DBASE.....	23

CAPÍTULO III	25
3.1 Metodologia	25
3.2 Replicação como Solução do Problema da Distribuição dos Dados	25
3.3 Métodos de Replicação de Dados	26
3.4 Snapshot Replication.....	27
3.5 Transactional Replication	31
3.6 Merge Replication.....	34
3.7 Problemas do Contexto de Banco de Dados Distribuído	38
CAPTÍTULO IV	39
4.1 Atual Sistema Administrativo da UNIPAR.....	39
4.2 Proposta de Replicação de Dados para a UNIPAR.....	40
4.2.1 Acesso às informações.....	40
4.3 Arquitetura Atual	41
4.4 Principais Tabelas	43
4.5 Servidores Envolvidos	44
4.6 Proposta de distribuição de dados	44
4.7 Análise da Proposta.....	45
4.8 Validar a Proposta.....	46
CAPÍTULO V	47
5.1 Conclusão.....	47
REFERÊNCIAS BIBLIOGRÁFICAS	48

ANEXOS	50
Anexo I.....	50
6.1 Configurando o SQL Server para Replicação com o método Merge.....	50
6.1.1 Criando uma Replicação Merge	51
6.1.2 Criando Subscribers para a Publicação	56

Lista de Figuras

Figura 1 - Arquitetura do <i>Snapshot Replication</i>	23
Figura 2 - Arquitetura da <i>Transaction Replication</i>	24
Figura 3 - Arquitetura da <i>Merge Replication</i>	26
Figura 4 - Estrutura Organizacional da Universidade Paranaense	29
Figura 5 - Diagrama da Arquitetura Atual do Sistema	31

Lista de Tabelas

Tabela 1 - Exemplo de Tabelas do Banco de Dados	16
Tabela 2 - Exemplo de Tabelas do Banco de Dados	16
Tabela 3 - Distribuição de Dados do Sistema Administrativo da UNIPAR.....	33

Lista de Abreviaturas

SQL 2000	Microsoft SQL Server 2000
WIN 2000	Microsoft Windows 2000
UNIPAR	Universidade Paranaense
BD	Banco de Dados
BDD	Banco de Dados Distribuídos
SGBD	Sistema de Gerenciamento de Banco de Dados
SGBDD	Sistema de Gerenciamento de Banco de Dados Distribuídos

Resumo

Palavras-chave: sistemas administrativos, banco de dados, replicação de dados.

Este trabalho visa apresentar uma solução de gestão de um sistema de informação administrativo através de métodos de replicação de dados entre instituições. Esta proposta vai tratar na instituição UNIPAR – onde treze unidades estão disponíveis entre setes cidades. A proposta estuda o método atual de distribuir seus dados, mostrando problemas decorrentes da utilização de bases de dados como DBASE em seus sistemas administrativos.

Abstract

Key-Words: administrative systems, database, replication of data.

This work to present a solution of administration of an administrative system of information through methods of replication of data among institutions. This proposal will treat in the institution UNIPAR - where thirteen units are available among seven cities. The proposal studies the current method of distributing their data, showing current problems of the use of bases of data like DBASE in their administrative systems.

CAPÍTULO I

1.1 Introdução

A evolução da tecnologia de banco de dados tem acompanhado quase sempre as necessidades das organizações em obter informações. De acordo com a demanda por informações, a tecnologia de banco de dados é uma ferramenta indispensável ao contexto organizacional.

Em banco de dados distribuídos, os dados são armazenados em locais diferentes, sendo que cada local é gerenciado por um sistema gerenciador de banco de dados que possui autonomia própria, ou seja, ele pode executar operações locais independentemente dos outros. Apesar da existência de várias possibilidades em distribuir dados, a visão clássica é que o contexto distribuído suporte duas propriedades: a independência dos dados distribuídos e a atomicidade das transações distribuídas. Dessa maneira, para o usuário do sistema de bancos de dados distribuídos, as operações são transparentes, pois ele não percebe quando e onde elas ocorrem.

A replicação de dados é abordada por um conjunto de tecnologias que permite a manutenção de várias cópias de dados idênticas em múltiplos locais. A replicação de dados é uma forma de distribuição de dados.

Dentre as várias possibilidades do uso de aplicações de banco de dados, este trabalho pretende responder a seguinte questão: Como os métodos de replicação de dados podem apoiar um sistema administrativo de uma Universidade multi-campi? Cada unidade possui sua autonomia e seus próprios dados. Todavia, a integração dos dados se faz necessária, principalmente para a administração financeira e acadêmica dessa universidade. Quanto à integridade desses dados, os sistemas gerenciadores deverão cumprir as restrições, tanto local quanto global, ou seja, se um aluno efetua o pagamento de sua mensalidade na tesouraria da instituição, nesse momento a baixa desse pagamento deve refletir globalmente, pois eventualmente outra pessoa pode querer efetuar o mesmo pagamento em outra unidade. Este é um simples exemplo, o qual deve ser observado para um sistema distribuído de banco de dados acadêmico.

1.2 Objetivo Geral

Objetivo geral deste trabalho é apresentar uma solução de gestão de um sistema de informação administrativo através de métodos de replicação dados.

1.3 Objetivo Específico

- Estudar os SGBD'S que oferecem mecanismos de replicação de dados;

- Estudar os métodos de replicação de dados do Microsoft SQL Server 2000 (SQL 2000);
- Estudar o sistema de um sistema de informação administrativo na UNIPAR;
- Apresentar uma proposta de replicação para o sistema de informação da UNIPAR;
- Validar a proposta apresentada.

1.4 Motivação

Com o passar do tempo novas tecnologias são desenvolvidas e um número maior de informações é gerado em tempos cada vez menores, e neste meio tempo, as tecnologias que já existiam irão se tornar obsoletas.

Porém, vários problemas de hardware ou de software deverão ser resolvidos antes que esta tecnologia de bancos de dados possa ser utilizada em sua totalidade. Alguns destes problemas são os mesmos do uso de bancos de dados distribuídos, porém não se devem confundir as duas tecnologias. Embora cada um deles com suas peculiaridades, entre as dificuldades em comum da implantação de bancos de dados distribuídos, estão a replicação de dados (fragmentos de dados armazenados em mais de uma localidade), os modelos de transações e processamento de consultas, a modelagem da base de dados e a linguagem de consulta a essas bases devem ter diferenças das comuns, tolerância à falhas e recuperação, entre outras.

Tendo em vista a necessidade de uma constante atualização, as empresas necessitam se manter em dia com suas tecnologias, assim a instituição de ensino UNIPAR, utilizando atualmente em seus sistemas administrativos, um sistema desenvolvido com tecnologia Clipper, com banco de dados DBase, seu sistema administrativo precisou estar distribuído entre suas várias unidades, para que os pagamentos, saldo em conta, solicitação de provas substitutivas e demais transações, fossem disponibilizadas entre as unidades de forma que estes estariam disponíveis a todas. Porém foram identificados problemas na distribuição destes dados, como falhas freqüentes e perda dos índices das informações transmitidas pela rede, dos quais não foi possível fazer sincronia entre os mesmos, pois o Clipper e o DBase não dispõem do recurso de distribuição dos dados, deixando esta função ao programador, que é susceptível a falhas.

Com a ocorrência destes problemas, uma nova tecnologia foi adotada para distribuição dos dados, utilizando métodos de replicação de dados. O SGBD utilizado é o Microsoft SQL Server 2000, usando um software com tecnologia Object Pascal (Delphi 8) e sistema operacional Microsoft Windows 2003 Server para os servidores e o Microsoft Windows 2000 Professional.

1.5 Justificativa

Atualmente, a tecnologia da informação envolve direta ou indiretamente todas as camadas da sociedade, sobretudo as organizações que podem agregar valores às suas atividades fazendo uso dessa tecnologia. Entretanto, a adoção dessa tecnologia requer conhecimento do ambiente no qual será inserida, bem como dos artefatos envolvidos (*hardware, software e telecomunicação*). As redes de computadores e os bancos de dados proporcionam a troca de informações em parcelas pequenas de tempo, tornando cada vez mais ágil a disseminação da informação na organização e/ou entre organizações.

Bancos de dados distribuídos permitem a integração de dados alocados em vários locais de um sistema distribuído, cada um com autonomia própria. Dentre as possibilidades de distribuição e replicação de dados, a principal preocupação é com a integridade dos mesmos. Normalmente, cada base de dados distribuída compõe um esquema global e, se uma dessas bases não mantém a integridade de seus dados, então todo o ambiente estará comprometido. Cada nova instância de qualquer banco de dados que forma o ambiente distribuído, deverá possuir consistência compatível aos demais. Daí a necessidade de verificarmos como os SGBD's tratam essa questão, considerando a definição e o cumprimento das restrições de integridade. O que requer atenção para a forma como o SQL 2000 implementa a replicação de dados.

1.6 Apresentação do Trabalho Desenvolvido

Este trabalho está estruturado de forma que ao ler os capítulos sequencialmente se obtenha os conceitos necessários à compreensão do domínio tratado.

Assim sendo, os se seguirão com as seguintes características:

Capítulo II - Contribui para o esclarecimento sobre os sistemas administrativos, bancos de dados, replicação de dados e suas aplicações nos sistemas administrativos de uma universidade multi-campi.

Capítulo III – Metodologia desenvolvida, métodos de replicação.

Capítulo IV – Atual arquitetura, aplicando os métodos de replicação.

Capítulo V – Conclusão.

Anexo I – Manual de utilização, e a aplicação do Método de Replicação de dados *Merge*.

CAPÍTULO II

2.1 Bancos de Dados

Um banco de dados consiste, conforme [ELMASRI 2000], em uma coleção de dados relacionados que representam alguns aspectos do mundo real. E o sistema gerenciador do banco de dados, é o software projetado para assistir a manutenção e utilização dos mesmos [RAMKRISHIMAN 2000].

Historicamente, a tecnologia de banco de dados surgiu nos anos 50 com os sistemas de arquivos de dados simples, sendo os mesmos acessados seqüencialmente. Mas a idéia de integração entre a base de dados e o software de gerenciamento aconteceu no início dos anos 60, desenvolvida por Charles Bachman na *General Eletronic*, e denominado *Integrated Data Store*. As Tabelas 1 e 2 são mostradas como as informações em uma estrutura de banco de dados ficam armazenadas. Essa idéia serviu como base para o desenvolvimento do modelo *network data model* (modelo em rede), padronizado pela *Conference on Data Systems Languages* (CODASYL). Mais tarde, a IBM – *International Busine Machine* desenvolveu o *Information Management System* (IMS precursor do *DataBase Management System* (DBMS) usando hoje na maioria das instalações de banco de dados. O IMS serviu como alternativa para a representação de um *framework* de dados, chamado *hieraquical data model* (modelo hierárquico). Nos anos 70, também a IBM, Edgard Codd propôs uma nova representação chamando-a de *relational data base* (modelo relacional)

[RAMKRISHIMAN 2000]. Desde então, a popularidade do modelo relacional consolidou-se nas organizações, devido aos avanços na tecnologia de microprocessadores e de redes de computadores locais, permitindo a troca de informações em parcelas de tempo muito pequenas.

Nome da Tabela		Cliente		
Chave	Nome	Tipo	Autoincremento	Obrigatório
PK	Id	Inteiro	Sim	Sim
	nome	varchar(60)	Não	Sim
	endereco	varchar(100)	Não	Não
	bairro	varchar(40)	Não	Não
	Cep	Inteiro	Não	Não
	cidade	varchar(60)	Não	Não
	estado	char(2)	Não	Não
	telefone	Inteiro	Não	Não

Tabela 1 - Exemplo de Tabelas do Banco de Dados

Nome da Tabela		Agenda		
Chave	Nome	Tipo	Autoincremento	Obrigatório
PK	Id	Inteiro	Sim	Sim
FK	id_cliente	Inteiro	Não	Sim
	Horário	Time	Não	Sim
	Data	Date	Não	Sim
	Descrição	Blob	Não	Sim

Tabela 2 - Exemplo de Tabelas do Banco de Dados

2.1.1 PK – Chave primária (Primary Key)

O Conceito de “Chave Primária” é fundamental para o correto entendimento de como funciona um Banco de Dados baseado no modelo relacional.

Por exemplo, se defino um campo “Número da Identidade”, da tabela Cliente, como sendo um campo do tipo Chave Primária, estou dizendo que não podem ser cadastrados dois clientes com o mesmo valor no campo "Número da Identidade". Na prática estou garantindo que não possam ser cadastrados dois clientes com o mesmo “Número de Identidade”.

Em outras palavras poderíamos dizer que o Campo Chave Primária identifica de Maneira Única cada Registro de uma Tabela, isto é, de posse do valor da Chave Primária somente localizaremos um registro com aquele valor no campo Chave Primária. Abaixo alguns exemplos de campos que podem ser definidos como chave primária:

- Campo CPF em uma tabela de cadastro de clientes.
- Campo CNPJ em uma tabela de cadastro de fornecedores.
- Matrícula do aluno em uma tabela de cadastro de alunos.
- Código da Peça em uma tabela de cadastro de peças.
- Matrícula do funcionário em uma tabela de cadastro de funcionários.
- Número do pedido em uma tabela de cadastro de pedidos.

2.1.2 FK – Chave estrangeira (Foreign Key)

É uma chave formada pela chave primária de outra tabela e a chave de um campo da tabela que recebe o relacionamento. Define um relacionamento entre as tabelas e pode ocorrer repetidas vezes.

2.2 Sistemas de Gerenciamento de Bancos de Dados

Os sistemas de gerenciamento de bancos de dados, que também são chamados de SGBD's, são sistemas que controlam ou como o próprio nome já diz, gerenciam os bancos dados. A maior parte dos bancos de dados existentes hoje, já oferecem um SGBD's para sua gerencia.

2.3 Importância dos SGBD's nos Sistemas Administrativos

Os Sistemas de Gerenciamento de Banco de Dados (SGBD's) são utilizados em sistemas administrativos proporcionando um ambiente tanto *conveniente* quanto *eficiente* para a recuperação e armazenamento das informações do banco de dados [SILBERSCHATZ 1999]. Os SGBD's facilitam a entrada de dados como também,

alterações, consultas, formulários, e o uso adequado destas informações para as organizações são importantes para que se tenha um banco de dados organizado.

Os dados das organizações podem ser considerados recursos porque uma mesma informação ou dado possuem uma duração e importância e este pode ser requisitado por várias partes da empresa, com isso torna-se importante gerenciá-los.

2.4 Fundamentos dos Sistemas Administrativos

A importância dos sistemas administrativos é levada em consideração, pois estes são os alicerces das organizações porque visam às organizações na manipulação dos dados institucionais. Os sistemas administrativos são necessitados a qualquer tipo de organização e aplicação, em diferentes áreas. A forma que uma organização trabalha, faz dela ser o que esta é, define o mecanismo de funcionamento da mesma, para que este tenha sempre, agilidade, qualidade no serviço, etc.

2.5 Sistemas Administrativos com DBASE

Atualmente ainda existem sistemas administrativos que utilizam esta tecnologia como solução para banco de dados, mas tendo em vista o crescimento das informações

existentes nas instituições, mais computadores estão conectados a estes bancos de dados e seus aplicativos começam a gerar problemas, dos quais podemos citar:

- Não foi encontrado no DBASE, mecanismos de lock e unlock (bloqueio e desbloqueio) em suas tabelas ou tuplas, para que um usuário na rede possa ter acesso exclusivo a uma determinada informação, assim ficaria difícil dizer quem tem prioridade na rede, para leitura ou gravação;
- Quando o terminal de um usuário solicitar uma requisição ao banco de dados, de tamanho 10Mbytes, por exemplo, sem especificar uma condição específica, os 10Mbytes trafegarão pela rede do servidor até o terminal. Esta transferência volumosa pode gerar colisões na rede e baixo rendimento;
- Se sua rede tem um padrão menor que 10Mbps, e dependendo da quantidade de informações utilizada pelo sistema em uso, é possível que dados sejam perdidos, e seu programa se tornará lento.

As possibilidades a cima, podem ser resolvidas com algumas formas de programação, mas como os programas são desenvolvidos por seres humanos, falhas e brechas nos sistemas poderão aparecer. Sendo assim, existe a necessidade de deixar esta função a cargo do SGBD's, em que em sua maioria, já oferece os recursos descritos à cima.

CAPÍTULO III

3.1 Metodologia

Os problemas de distribuição dos dados serão resolvidos utilizando dois ou mais computadores equipados com processadores Intel ou AMD similar ou superior a 1.0Ghz com 512Mb de memória RAM, cada um destes sendo um *site* com o Microsoft SQL Server 2000 conforme a necessidade da instituição. O sistema operacional utilizado será o Microsoft Windows 2000 Server (a partir desse ponto será referenciado como SQL 2000), por ser um sistema estável e também por oferece boas ferramentas para administração de servidores.

3.2 Replicação como Solução do Problema da Distribuição dos Dados

A replicação de dados será utilizada na instituição como a solução para sanar seus problemas de distribuição dos dados. O método escolhido para replicação de dados foi a *merge replication*, método este de replicação assíncrona, ou seja, os dados são replicados em um horário específico ou de tempo em tempo, conforme a necessidade.

3.3 Métodos de Replicação de Dados

A replicação no SQL 2000 é abordada por um conjunto de tecnologias que permite a manutenção de várias cópias de dados idênticas em múltiplos *sites*. No entanto o modelo de replicação (distribuição de dados) adotado pelo SQL 2000 é o *publisher-subscriber* composto pelo *publisher*, *subscriber* e *distributor*:

Um *publisher* é o servidor que origina os dados a serem replicados. Ele permite um item para cada relação ou um outro objeto (view, stored procedure) do banco de dados para ser usado na origem da replicação. Um ou mais itens relacionados no mesmo banco de dados são organizados em uma publicação.

Publicações são caminhos para grupos de dados e objetos que se deseja replicá-los [WEB 2003a].

O *subscriber* é um servidor que recebe os dados replicados pelo *publisher*. O *subscriber* define uma subscrição (espécie de assinatura) para uma particular publicação [WEB 2003a].

O *distributor* é um servidor que realiza várias tarefas, principalmente a de ser mediador quando os itens são movidos do *publisher* para *subscriber*. Essas tarefas dependem do tipo de replicação adotada.

O SQL 2000 realiza três tipos de replicação: *snapshot replication*, *transacional replication* e *merge replication*. Também suporta replicações originadas de dados heterogêneos através dos protocolos OLE DB ou ODBC que pode subscrever publicações no SQL 2000. O SQL 2000 também pode receber dados replicados de várias origens como: Microsoft Exchange, Microsoft Access, Oracle e DB2.

3.4 Snapshot Replication

A *snapshot replication* copia instantaneamente dados ou objetos do banco de dados exatamente como eles existem. As publicações *snapshot* são tipicamente definidas para acontecerem em bases programadas. O *subscriber* contém cópias dos itens publicados à medida que eles aparecem na última *snapshot*. A *snapshot replication* é usada onde a origem do dado é relativamente estática.

A *snapshot replication* é implementada por dois agentes: *snapshot agent* e *distribution agent*. O *snapshot agent* prepara os arquivos *snapshot* contendo: esquema, dados das relações publicadas e objetos armazenados em arquivos *snapshot folder*. Em seguida registra a sincronização dos *jobs* no banco de dados de distribuição (*distributor*). Por *default* o *snapshot folder* é localizado no *distributor*, mas pode ser especificado em localizações alternadas em vez de usar a localização *default* [WEB 2003b]. O *distribution agent* move *snapshot* armazenado nas relações do banco de dados de distribuição, para as relações de destino nos *subscriber*. O banco de dados de

distribuição é usado somente pela replicação e não contém nenhuma relação definida pelo usuário.

A cada momento o *snapshot agent* é executado, ele verifica se qualquer nova subscrição foi adicionada. Se não há novas subscrições, nenhum script, ou arquivos de dados são criados. Se a publicação é gerada com a opção para criar imediatamente o primeiro *snapshot*, então novos esquemas e arquivos de dados são criados no momento em que o *snapshot agent* é executado. Todos os esquemas e arquivos são armazenados no *snapshot folder*, portanto o *distribution agent* ou *merge agente* os transfere para *subscriber*. Essa transferência ainda pode ser efetuada manualmente. O *snapshot agent* realiza os seguintes passos [WEB 2003b]:

- 1 - estabelece uma conexão do *distributor* para o *publisher* e prepara *share-locks* em todas as relações inclusas na publicação. O *share-lock* garante uma consistência *snapshot* de dados. Porque os *locks* não permitem que outros usuários atualizem as relações, o *snapshot agent* deve ser programado para executar quanto a atividade do banco de dados estiver baixa.
- 2 – estabelece uma conexão do *publisher* para o *distributor* e esteve uma cópia do esquema da relação para cada item para arquivo (.sch). Se houver a inclusão de índices e declarativas de integridade referencial, o script do agente de saída seleciona os índices para um arquivo (.idx). Outros objetos do banco de dados. Como *stored procedure*, *views*, funções definidas pelo usuário,

entre outros, também podem ser publicados como parte da replicação.

- 3 – copia os dados das relações publicadas no *publisher* e os escreve no *snapshot folder*. Se todos os *subscribers* são instâncias do SQL Server, o *snapshot* é armazenado como volume nativo. Se um ou mais *subscriber* é uma origem de dados heterogêneos, o *snapshot* é armazenado em um arquivo no modo caractere.
- 4 – anexar linhas para o *Msrepl_commands* e o *Msrepl_transactions* nas relações no banco de dados de distribuição. As entradas das relações *Msrepl_commands* são comandos indicando a localização do conjunto de sincronização (arquivos *.sch* e *.bcp*) e referências para qualquer script de pré-criação especificada. As entradas nas relações *Msrepl_transactions* referenciam o *subscriber* da tarefa de sincronização.
- 5 – libera os *share-locks* de cada relação publicada e finaliza escrevendo os históricos de *logs* na relação *Msrepl_transaction*.

Por sua vez o *distribution agent* realiza os seguintes passos [WEB 2003b].

- 1 – estabelece uma comunicação entre o servidor (em que o *agent* está localizado) e o *distributor*. Para mandar as subscrições. O

distribution agent é executado normalmente no *distributor* e para obter as subscrições *distribution agent* é executado no *subscriber*.

- 2 – examina as relações *Msrepl_commands* e *Msrepl_transactions* no banco de dados de distribuição no *distributor*. O agente aponta a localização da sincronização da primeira relação e os comandos de sincronização do *subscriber* junto das tabelas.
- 3 – aplica o esquema e comandos para a subscrição do banco de dados. Se o *subscriber* não é uma instância do SQL 2000, o agente converte os tipos de dados conforme a necessidade. Todos os itens de publicação são sincronizados, preservando a transação e a integridade referencial entre as relações bases.

A Figura 1 mostra a arquitetura da replicação do tipo *snapshot*, na qual os agentes se cooperam para efetuar a replicação desse tipo.

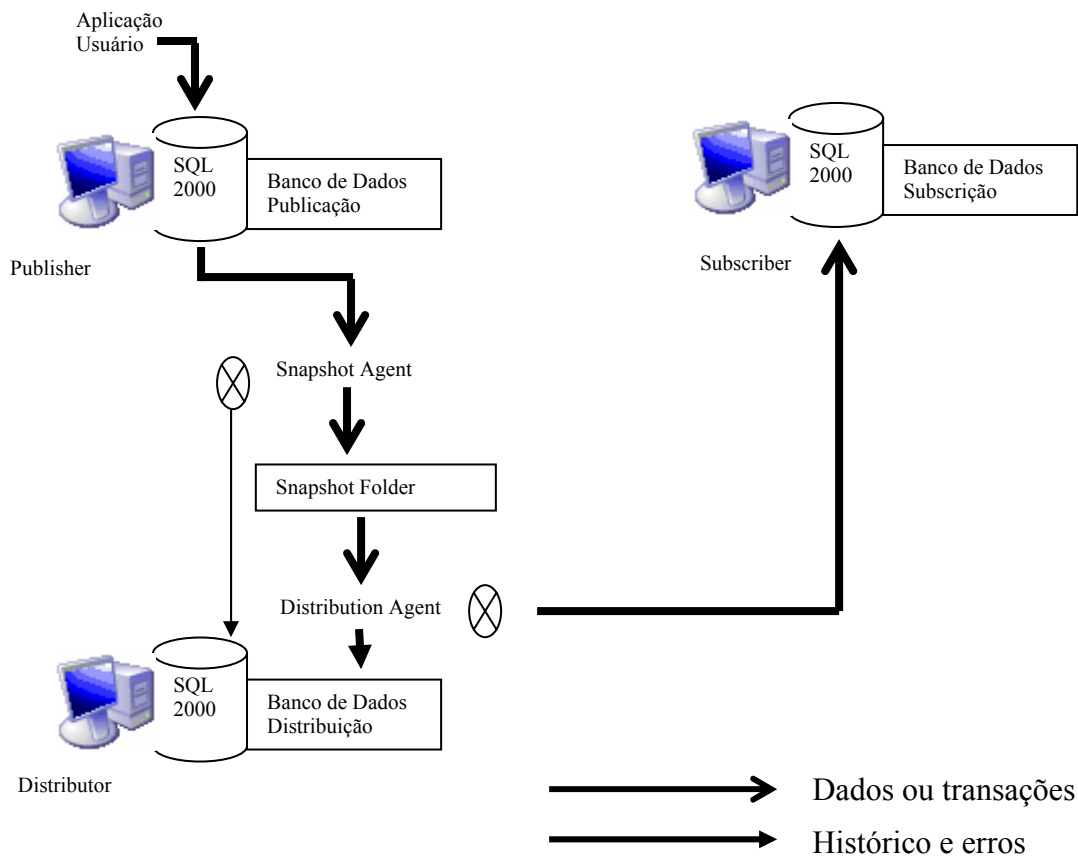


Figura 1 - Arquitetura do *Snapshot Replication*

3.5 Transactional Replication

Na *transactional replication* os *subscribers* são sincronizados primeiro com o *publisher*, tipicamente usando um *snapshot*, então as modificações de dados e transações são capturadas e enviadas para o *subscriber*. A integridade transacional é mantida nos *subscribers* por ter todas as modificações feitas no *publisher*, que são replicadas para *subscriber*. A replicação transacional é usada quando os dados devem ser replicados e modificados, sendo assim a transação deve ser preservada. Os

publishers e os *subscribers* são confiáveis, pois frequentemente estão conectados em rede [WEB 2003c].

A *transactional replication* é implementada pelos seguintes agentes: *snapshot agent*, *log reader agent* e *distribution agent*. O *snapshot agent* prepara os arquivos contendo esquema, dados das relações publicadas e os objetos do banco de dados, armazena-os em um arquivo no *snapshot folder* e ainda registra *jobs* de sincronização no banco de dados de distribuição no *distributor*.

O *log reader agent* monitora o *log* da transação para cada banco de dado configurado para replicação transacional. Copia as transações marcadas para a replicação do *log* de transação no banco de dados de distribuição. O *distribution agent* move os *jobs snapshot* e as transações seguras nas relações do banco de dados distribuição para *subscribers* [WEB 2003c].

Este tipo de replicação (Figura 2) permite atualizações nas transações de dados marcadas para a replicação, então os mecanismos da *transactional replication* devem ser analisados com mais rigor devido a complexidade inerente a esse tipo de replicação.

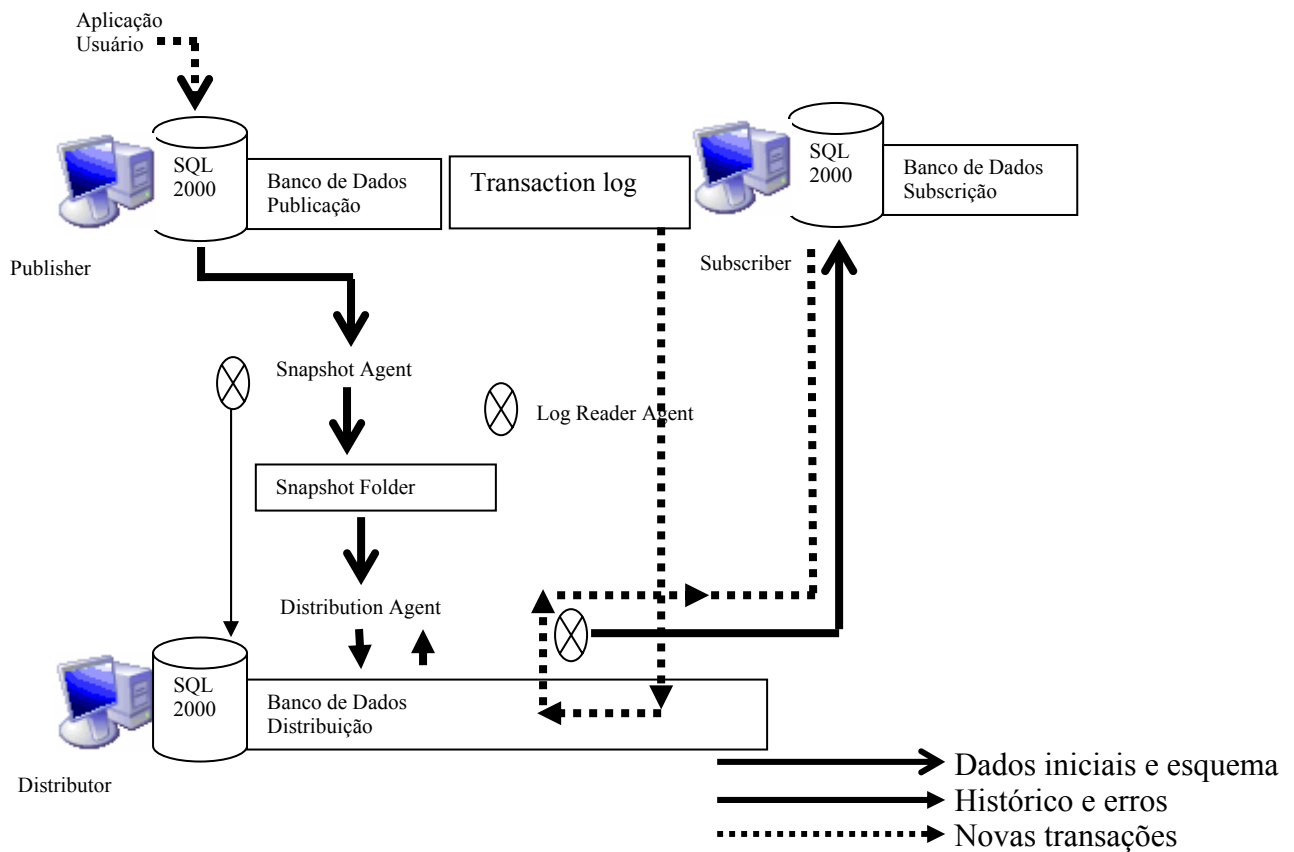


Figura 2 - Arquitetura da *Transaction Replication*

Antes do *subscriber* receber a *transactional replication*, o *publisher* pode sofrer alterações incrementais, mas o *subscriber* deve conter as relações com o mesmo esquema e dados como os do *publisher*. A cópia completa da publicação corrente do *publisher* para o *subscriber* é chamada de aplicação do *initial snapshot*. Quando *snapshot* são distribuídos e aplicados nos *subscribers*, somente aqueles *subscribers* que esperaram pelo *initial snapshot* são afetados, enquanto que outros *subscribers* não são [WEB 2003c].

Enquanto *snapshot* é gerado, o SQL 2000 normalmente aplica *share-locks* em todas as relações publicadas como parte da replicação. Isto previne as atualizações das relações na publicação.

Concorrência com processamento *snapshot* é disponível somente com *transactional replication*. Mas durante as entradas para geração do *snapshot*, os *share-locks* não aparecem, permitindo aos usuários continuarem trabalhando criando seus arquivos *initial snapshot* normalmente.

3.6 Merge Replication

A *merge replication* permite a múltiplos *sites* trabalharem independente com um conjunto de *subscribers*, portanto uma fusão combinada de trabalho volta para *publisher*. Os *subscribers* e *publishers* são sincronizados como um *snapshot*. Modificações são rastreadas para uma nova versão de dados. Durante a junção, alguns conflitos podem ser encontrados quando múltiplos *subscribers* modificam o mesmo dado.

A *merge replication* suporta algumas definições para a resolução de conflitos, as quais são conjuntos de regras que definem como resolver esses conflitos. *Custom conflict resolver script* pode ser escrito em qualquer lógica, o qual pode ser necessário para resolver conflitos considerado a complexidade do cenário do conflito. A *merge replication* é usada onde um *subscriber* opera de forma independente, ou quando vários *subscribers* podem atualizar o mesmo dado [WEB 2003c].

A *merge replication* é implementada por dois agentes: *snapshot agent* e *merge agent*. O *snapshot agent* prepara os arquivos *snapshot* contendo o esquema e os dados das relações publicadas, armazena os arquivos no *snapshot folder* e insere e sincronização dos *jobs* no banco de dados de publicação. O *snapshot agent* também cria replicações específicas, armazenando *procedures*, *triggers* e *system tables* (relações do sistema) [WEB 2003d].

O *merge agent* aplica o *initial snapshot jobs* nas relações do banco de dados de distribuição para os *subscribers*. Eles também agrupam as modificações incrementais de dados que ocorrem no *publisher* ou nos *subscribers* depois que o *initial snapshot* tenha sido criado e, reconcilia os conflitos de acordo com as regras configuradas ou escritas no *custom resolver*.

O papel do *distributor* (Figura 3) é muito limitado na *merge replication*, conseqüentemente a implementação do *distributor* é simples, pois o *distributor agent* não é usado durante toda a *merge replication*.

2000 usará essa coluna automaticamente como o identificador da linha para publicação da relação.

Os *triggers* quando utilizados na *merge replication* rastreia as modificações dos dados em cada linha ou em cada coluna, captura as modificações realizadas nas relações publicadas e registra as modificações ocorridas nas relações *merger system*. Diferentes *triggers* são gerados para rastrear os itens modificados no nível de coluna ou no nível de linha. Não há interferência de aplicações que utilizam *triggers* com a *merge replication*.

O *snapshot agent* também cria *stored procedure* customizadas que atualiza o *subscription database*. Também encontramos uma *stored procedured* customizada para os comandos de manutenção do *subscription database* (*insert, delete e update*). Quando o dado é atualizado e ou um novo registro precisa ser inserido no *subscription database* então as *storeds procedures* customizadas são usadas preferencialmente para os comandos (*insert, delete e update*) individualmente.

As *systems tables* são adicionadas no banco de dados para suportar o rastreamento de dados, sincronização eficiente, e detecção de conflito. Para qualquer linha criada ou modificada na relação, a relação *Msmerge_contents* contém a criação das mais recentes modificações ocorridas. Ela também possui a versão da linha, assim como para qualquer atributo dessa linha. A relação *Msmerge_tombstone* armazena os *delete's* dos dados contidos em uma publicação [WEB 2003d].

3.7 Problemas do Contexto de Banco de Dados Distribuídos

A adoção do banco de dados distribuídos pelas organizações concretiza a descentralização de seus negócios. Mas os fatores complicadores, aqueles encontrados no banco de dados centralizados, agora são multiplicados pelo número de *sites*, além de outros problemas caracterizados pelos sistemas gerenciadores de banco de dados distribuídos.

Os problemas característicos do banco de dados distribuídos são influenciados por três fatores principais: a) havendo replicação da base dados, ou parte dela, todas as entradas e/ou atualizações executadas devem refletir em cada elemento do dado replicado; b) se algum *site* falhar ou *link* de comunicação for interrompido (tornando um ou mais *sites* incomunicáveis) enquanto uma atualização é executada, o sistema deve contornar os reflexos da falha até que o sistema esteja restabelecido; c) considerando que cada *site* não tem informação instantânea ou as ações estão carregadas em outros *sites* (oposto do centralizado), então a sincronização das transações em múltiplos *sites* é considerado pesada [OZSU 2001].

CAPTÍTULO IV

4.1 Atual Sistema Administrativo da UNIPAR

O sistema atual de informação administrativo da UNIPAR é utilizado por várias unidades localizadas geograficamente em cidades diferente, como mostrado na Figura 4.

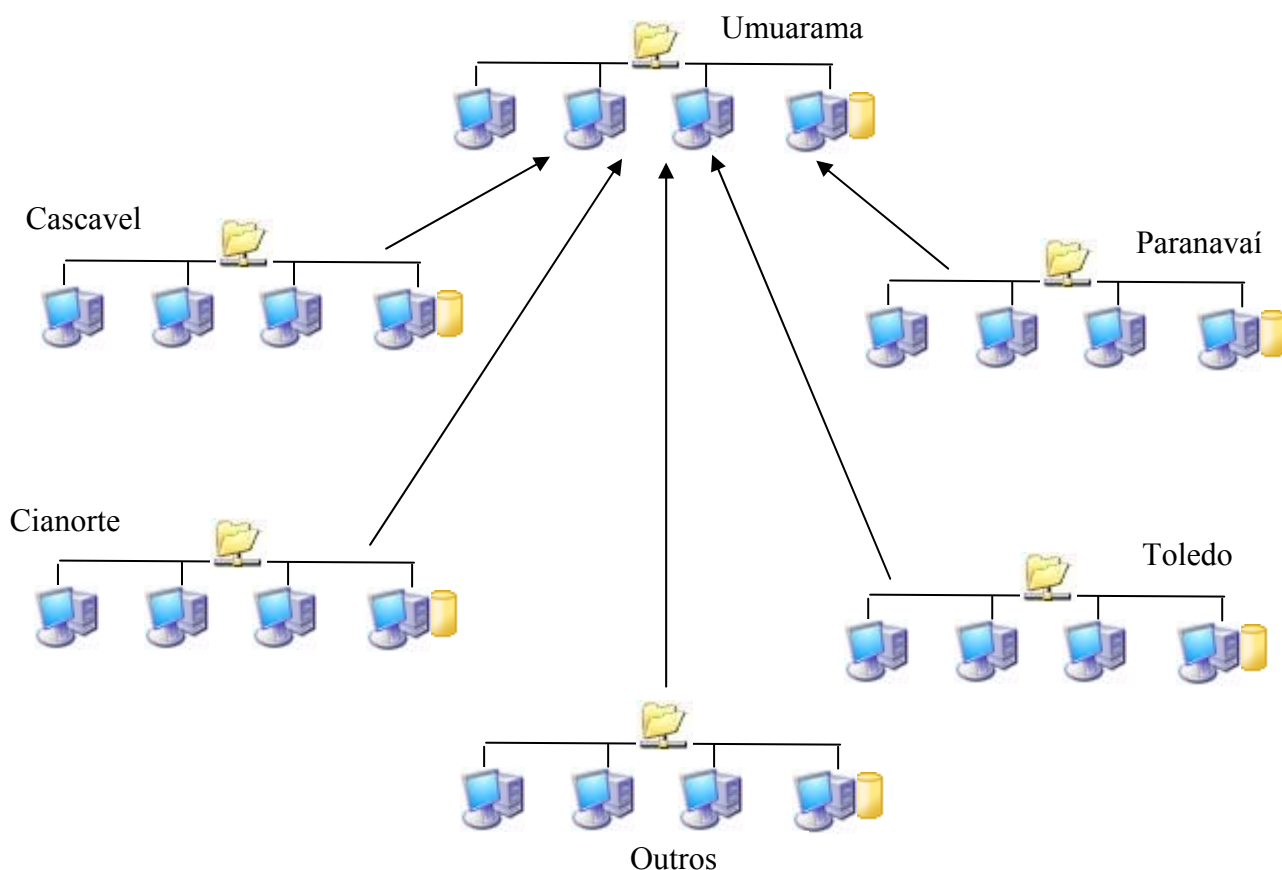


Figura 4 - Estrutura Organizacional da Universidade Paranaense

Cada uma das treze unidades espalhadas por sete cidades mantém um sistema administrativo, no qual ao término do dia os dados de cada unidade são enviados para a unidade central localizada no Campus Sede, na cidade Umuarama, sendo assim centralizados e contabilizados.

4.2 Proposta de Replicação de Dados para a UNIPAR

4.2.1 Acesso às informações

Os sistemas administrativos da UNIPAR são consultados por vários departamentos. Porém, as informações consultadas por cada departamento têm níveis de detalhamento distintos.

- A Tesouraria registra informações financeiras, bem como pagamentos de mensalidades;
- A Secretaria efetua a matrícula dos acadêmicos;
- A Diretoria necessita de todas as informações contratuais para o gerenciamento efetivo do mesmo;
- Os Recursos Humanos efetua a contratação de professores e funcionários administrativos.

Inicialmente, por falta de planejamento da complexidade em seu sistema administrativo, o esquema de banco de dados foi implementado de forma centralizada em um servidor em cada unidade. Este servidor tem grande capacidade de processamento e armazenamento, sendo um grande investimento realizado pela UNIPAR. Tendo em vista o crescimento do banco de dados, novos investimentos são realizados conforme a necessidade.

Entretanto, os usuários que acessam o sistema através das unidades exceto as situadas na cidade de Umuarama, só terão seus dados plenamente no sistema administrativo no dia seguinte, limitando assim que acadêmicos possam ter acesso a seus dados através de sistemas on-line, como consulta de notas e frequências.

4.3 Arquitetura Atual

A figura 5 mostra a arquitetura atual do sistema em apenas uma unidade.

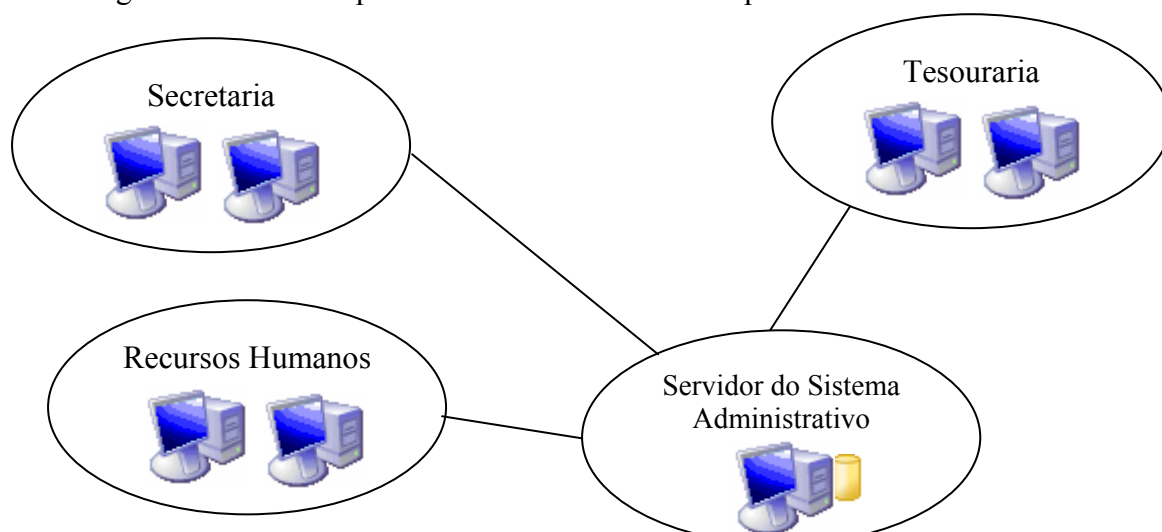


Figura 5 - Diagrama da Arquitetura Atual do Sistema

A configuração mínima do cliente para executar o sistema com bom desempenho é:

- Processador Intel ou similar com frequência de clock de no mínimo 100Mhz;
- 32Mb de memória *RAM* (*Randon Acess Memory* – Memória de acesso randômico);
- 10Mb de espaço disponível;
- Cliente Novell instalado e customizado.

A configuração atual do servidor onde está localizado o banco de dados centralizado do sistema é:

- Intel Pentium 4 Dual Xeon 3.06Ghz;
- Processador CISC;
- 1.000 MegaBytes de Memória *RAM*;
- Capacidade de armazenamento de 10Gb disponíveis para o banco de dados do sistema.

4.4 Principais Tabelas

O esquema físico de banco de dados do sistema é composto por 70 tabelas. As tabelas podem ser agrupadas de acordo com nível de acessibilidade e manutenibilidade, tendo em vista a facilidade das mesmas sobre os vários servidores da administração.

- Tabelas Principais – São tabelas de alta manutenção e acessibilidade, utilizadas por vários departamentos;
- Tabelas de Uso Geral – São tabelas de baixa manutenção e acessibilidade utilizada por vários departamentos;
- Tabelas de Controle dos Sistemas – São tabelas de baixa manutenção e acessibilidade utilizada como configuração do sistema;
- Tabelas de Uso Específico – São tabelas de alta acessibilidade e manutenção de uso específico por uma determinada empresa.

Para demonstrar o nível de acessibilidade e manutenibilidade do sistema, necessitamos conhecer o volume de informações destes grupos de tabelas que possuem alta acessibilidade e manutenibilidade e a respectiva taxa de crescimento anual.

Na administração direta são efetuadas, em média 6.000 (seis mil) matrículas anuais. Em cada matrícula existem pelo menos oito disciplinas para cada curso, para serem executadas, em média, por 13 unidades espalhadas em sete cidades. Isto reflete diariamente nas tabelas principais, que possuem em média 624.000 acessos anuais, só

para cadastramento, sem adicionar as possíveis consultas e projeções que podem ser executadas sobre as informações a qualquer momento.

Portanto, com base nos dados acima, é de fundamental importância a distribuição das tabelas a fim de reduzir o tráfego e melhorar a performance do sistema.

4.5 Servidores Envolvidos

Conforme descrito anteriormente, o sistema se encontra localizado no departamento de Desenvolvimento / Internet na unidade campus sede de Umuarama, exigindo do servidor grande capacidade de armazenamento e processamento das informações. Os outros servidores estão localizados nas outras unidades sendo um para cada cidade.

4.6 Proposta de distribuição de dados

Com base na distribuição das tabelas do sistema, seus níveis de acesso e da capacidade tecnológica instalada, pode-se utilizar os agrupamentos dos departamentos definidos anteriormente para estabelecer uma metodologia de distribuição de informações.

Os SGBD's das unidades deverão ser replicados para a unidade do campus sede de Umuarama para a centralização das informações da instituição. Isto é realizado de forma programada, para não sobrecarregar as linhas de comunicação, elegendo o servidor do campus sede de Umuarama como o servidor que coordenará a réplica e que estará sempre atualizado.

A Tabela 3 resume a forma de distribuição dos grupos de tabelas do sistema.

Equipamento	Grupo de Tabelas	Controle de Distribuição
Servidor Central	Principal	Máster
	Uso Geral	Máster
	Controle do Sistema	Máster
	Uso Específico	Máster
Servidores das outras Unidades	Principal	Merge – Atualização Diária
	Uso Geral	Merge – Atualização Diária
	Controle do Sistema	Merge – Atualização Diária
	Uso Específico	Merge – Atualização Diária

Tabela 3 - Distribuição de Dados do Sistema Administrativo da UNIPAR

4.7 Análise da Proposta

Com a distribuição do banco de dados do Sistema Administrativo da UNIPAR, tende-se a diminuir o tráfego das redes de transmissão de dados e a utilização de seus servidores, sendo possível assim aproveitar a capacidade disponível destes recursos para

utilização destes para outros meios. Em outros casos, o tráfego estaria concentrado nas redes locais, ocupando as linhas de comunicação e aumentando a concorrência pela banda da rede. Assim, obteríamos um desempenho melhor tanto para o sistema administrativo como para outras aplicações que estão se comunicando pela mesma.

4.8 Validar a Proposta

A proposta apresentada só será validada após a conclusão do término do desenvolvimento do sistema que acontecerá no mês de outubro de 2005 e ficará em fases de teste até dia 31 de dezembro de 2005, sendo assim, a validação deste trabalho só poderá ser apresentada posteriormente ao término desta data.

CAPÍTULO V

5.1 Conclusão

Neste trabalho apresentou-se uma proposta para implementação de replicação entre bancos de dados, partindo de uma combinação adequada de técnicas de replicação.

Não foram realizados experimentos em ambiente controlado, com métricas adequadas para a avaliação da proposta apresentada. Porém, um exemplo de implementação dessa solução foi apresentado, permitindo inferir, a partir dos resultados obtidos, que a proposta é viável, garante expansibilidade, portabilidade, baixo custo e ainda possibilita o uso conjunto de sistemas de gerência de banco de dados livres e proprietários.

A partir desta pesquisa, muito pode ser feito e alguns aspectos poderão ser implementados em trabalhos futuros, tais como:

a) realização de experimentos em ambiente controlado, a fim de mensurar o desempenho e a expansibilidade da replicação, quando implementada conforme sugestão contida neste trabalho;

b) utilização de replicação de dados em lugar da replicação procedural, o que permitiria a inclusão de recursos para detecção e tratamento de conflitos na solução proposta.

REFERÊNCIAS BIBLIOGRÁFICAS

[DATE 2000] DATE, C. J. **Introdução a Sistemas de Bancos de Dados**, trad. Vandenberg D. de Souza; Rio de Janeiro: Campus, 2000.

[ELMASRI 2000] ELMASRI, Ramez e NAVATHE. B. Shamkant **Fundamentals of Database System**. Third Edition, Addison-wesley. 2000 Pág. 4,74 – 100, 766 – 793.

[GUNDERLOY 2001] GUNDERLOY, Mike; L. JORDEN, Joseph, **Dominando SQL Server 2000**; São Paulo: Makron Books, 2001.

[LEAO 2000] LEAO, Renata de Oliveira; *Microsoft SQL Server 2000: estrutura e implementação de sistemas de banco de dados*; São Paulo: Érica , 2002.

[OZSU 2001] OZSU M. Tamer e VALDURIEZ Patrick “**Distributed and Parallel Database System**” second edition Prentice-Hall, Inc 1999 pág. 1 a 2001.

[RAMKRISHIMAN 2000] RAMKRISHIMAN R.: GEHRKE J. **Database Managemnet System**. Second Edition, 2000, McGraw-Hill Internationa Editions.

[SILBERSCHATZ 1999] SILBERSCHATZ, Abraham. **Sistema de Bancos de Dados**, trad. Marília Guimarães Pinheiro; São Paulo: Makron Books, 1999.

[WEB 2003a] MICROSOFT, **Replication Archicture**. Disponível em: http://msdn.microsoft.com/library/psdk/sql8_ar_ra_6u05.htm. Acesso em: 20/12/2003.

[WEB 2003b] MICROSOFT, **How Snapshot Replication Works**. Disponível em: http://msdn.microsoft.com/library/psdk/rpltypes_/2f8z.htm. Acesso em: 20/12/2003.

[WEB 2003c] MICROSOFT, **How Transaction Replication Works**. Disponível em: http://msdn.microsoft.com/library/psdk/rpltypes_/54j.htm. Acesso em: 20/12/2003.

[WEB 2003c] ICROSOFT, **How Merge Replication Works**. Disponível em http://msdn.microsoft.com/library/psdk/rpltypes_/52b.htm. Acesso em: 20/12/2003.

[WEB 2003d] ICROSOFT, **How Transaction Replication Works**. Disponível em: http://msdn.microsoft.com/library/psdk/rpltypes_/30z.html. Acesso em: 20/12/2003.

ANEXOS

ANEXO I

Configurando o SQL Server para Replicação com o método Merge

Antes de se replicar dados de qualquer um dos bancos de dados do seu SQL Server, é necessário configurar o ambiente para replicação. Esse processo criará a infraestrutura necessária (distribution database, SQL Server Agent Jobs, etc...) para o servidor armazenar as publicações e receber subscrições de outros servidores.

Configurando seu servidor com Publisher e Distributor:

- Usando o Enterprise Manager, inicie o “Configure Publishing and Distribution” Wizard – Procure a pasta “Replication” no seu servidor. Clique com o botão direito e selecione “Configure Publishing and Subscribers”
- Configure o seu servidor para que o mesmo seja seu próprio Distributor
- Você poderá ver durante o setup, uma informação dizendo que o serviço SQL Server Agent, está sendo executado sobre uma conta local do sistema. Isto não terá nenhuma implicação CASO o processo de replicação esteja sendo feito no mesmo servidor
- Configure o SQL Server Agent para iniciar automaticamente

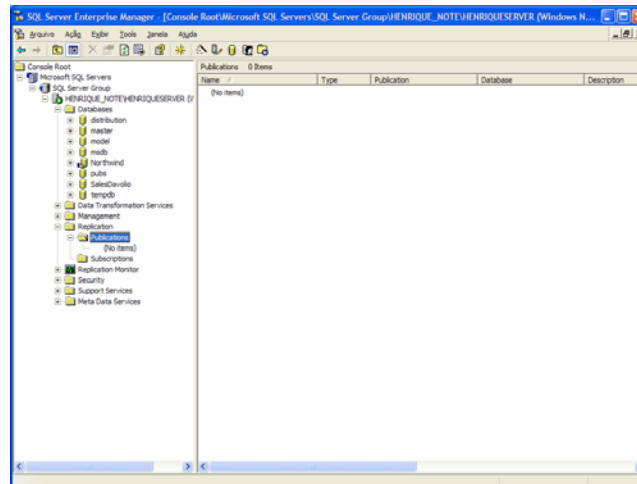
- Aceite as configurações padrão e finalize

Criando uma Replicação Merge

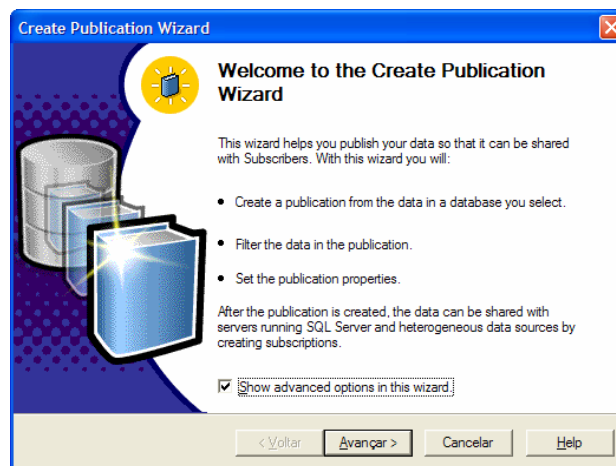
Utilizando o banco de dados Northwind, simularemos uma situação onde criamos uma força de vendas externa. Essa aplicação utilizará o recurso de Merge Replication do SQL Server 2000 para permitir que as pessoas no campo, possam manipular e atualizar os dados de vendas e cadastro de seus clientes, e periodicamente atualizar o banco de dados corporativo.

Crie uma publicação merge no database “Northwind” e faça o processo de sincronização de uma nova subscrição da força externa de vendas.

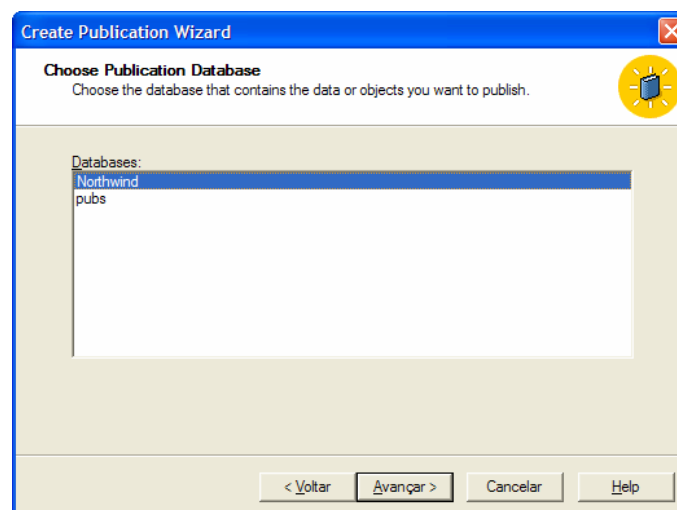
- Inicie o “Create Publication” Wizard no Enterprise Manager - Dentro da pasta “Replication”, voce verá uma pasta “Publications”, clique na mesma com o botão direito e selecione “New Publication”



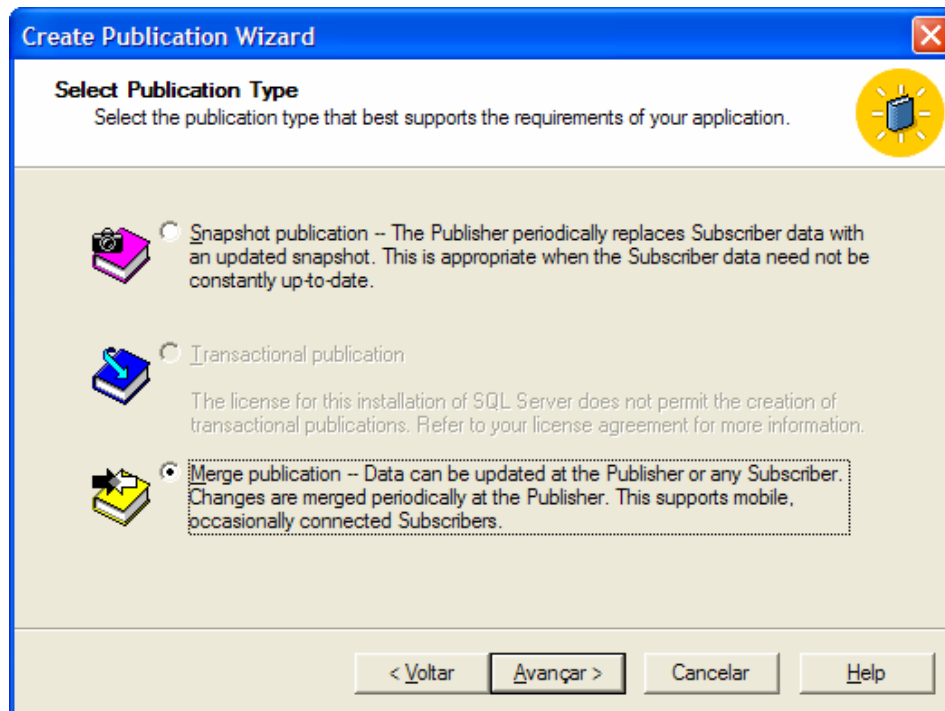
- Selecione a opção “Show Advanced Options” na tela inicial



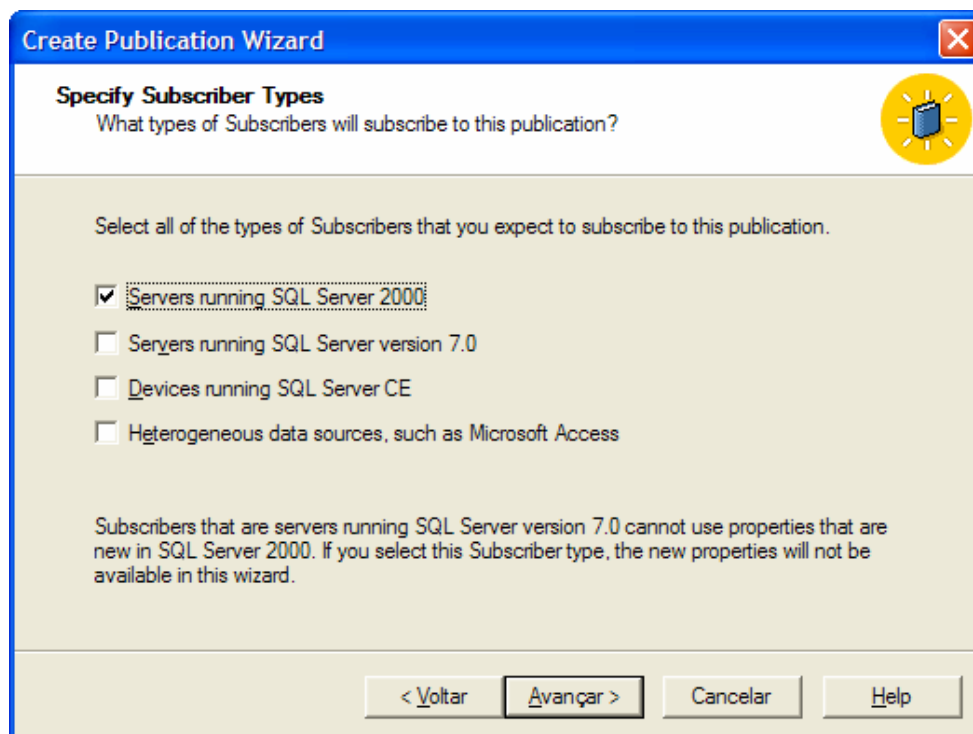
- Selecione “Northwind” como sendo o banco de dados de publicação



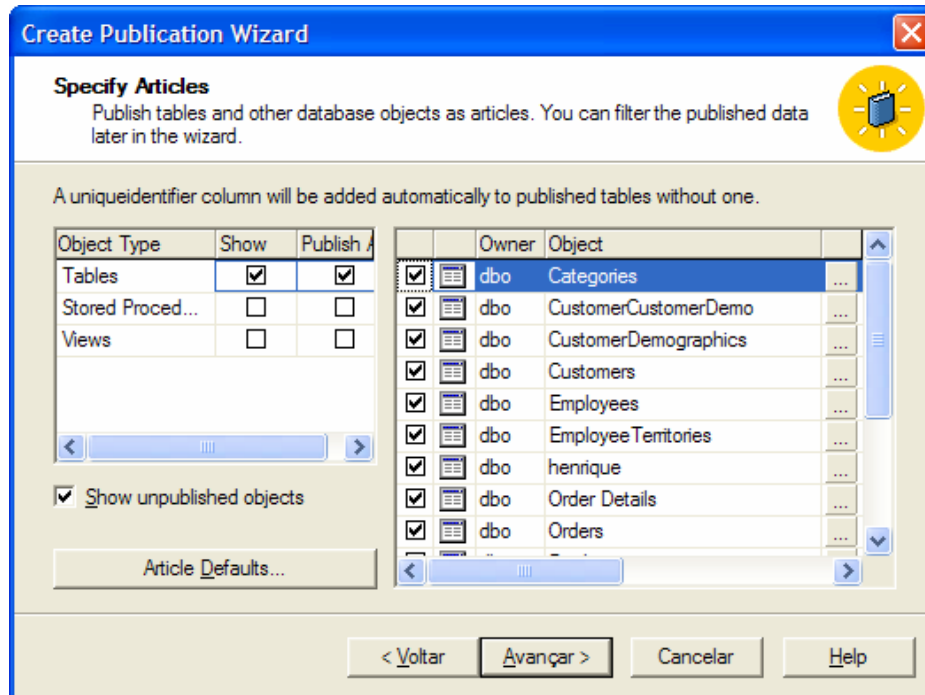
- Selecione “Merge”, como sendo o tipo de publicação



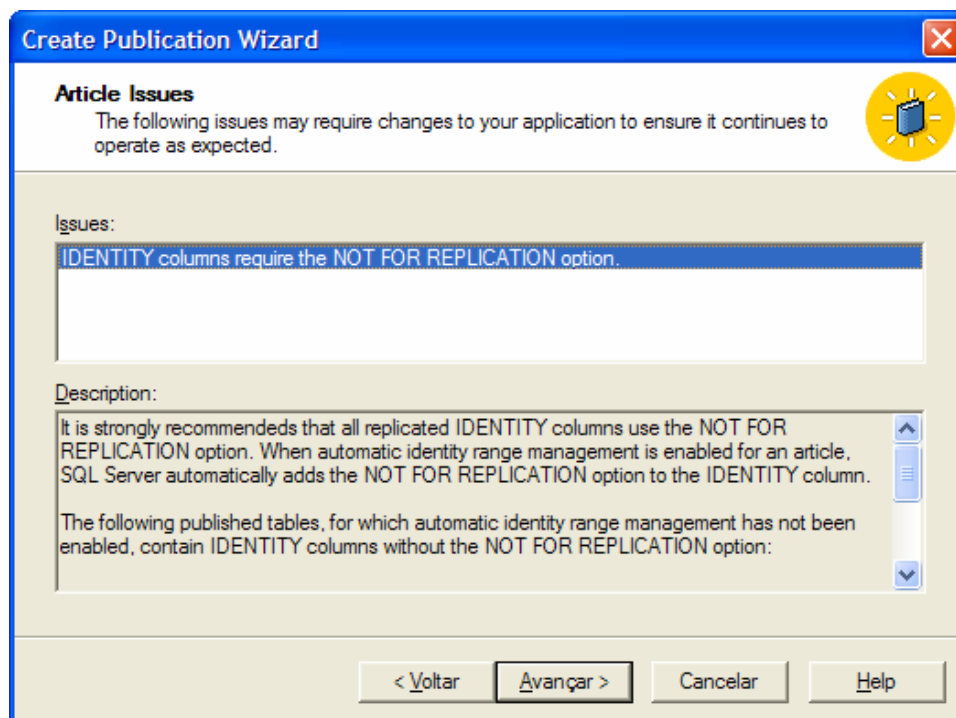
- Selecione SQL Server 2000, como sendo o tipo de Subscribers



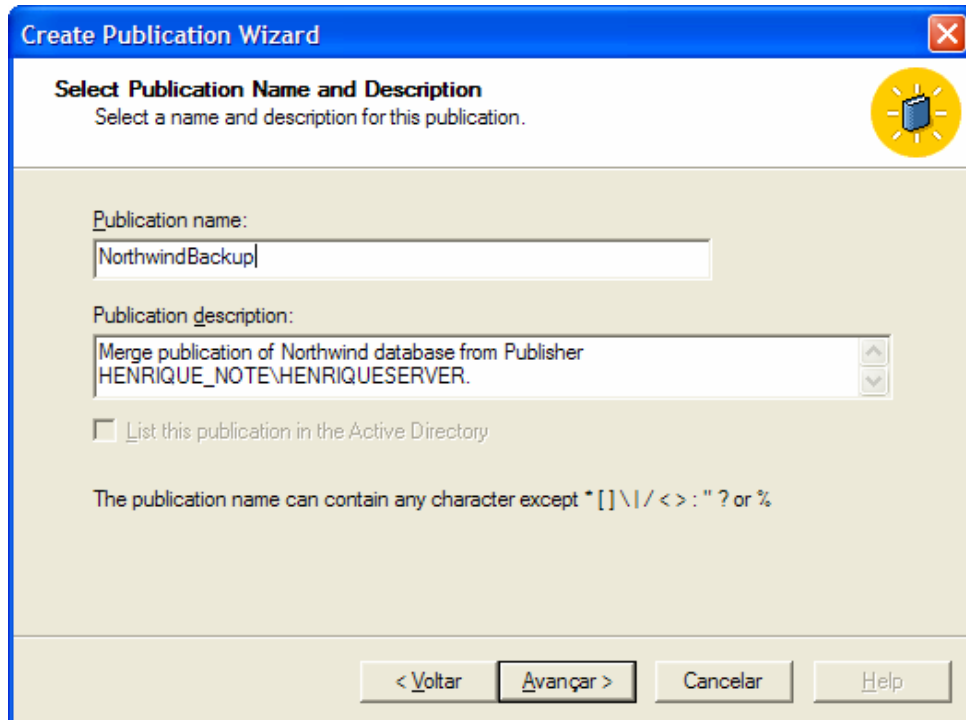
- Selecione todas as tabelas do banco de dados, como sendo os objetos a serem publicados



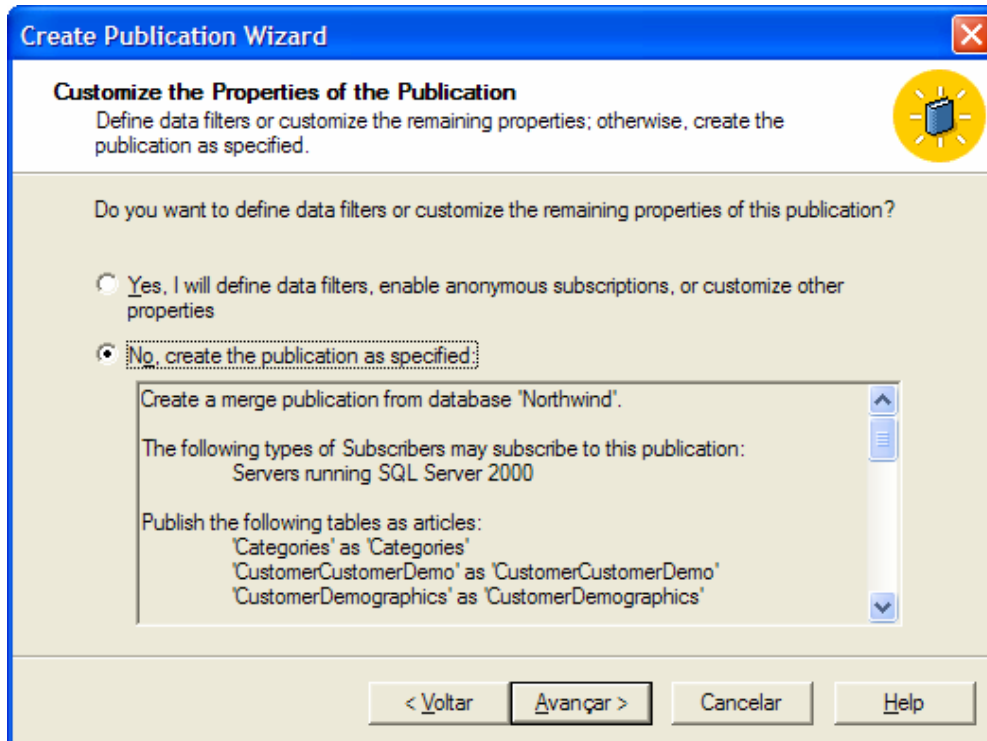
- Apenas clique em avançar



- Informe o nome da sua publicação



- Aceite as propriedades default de sua publicação



- E clique em Finish.

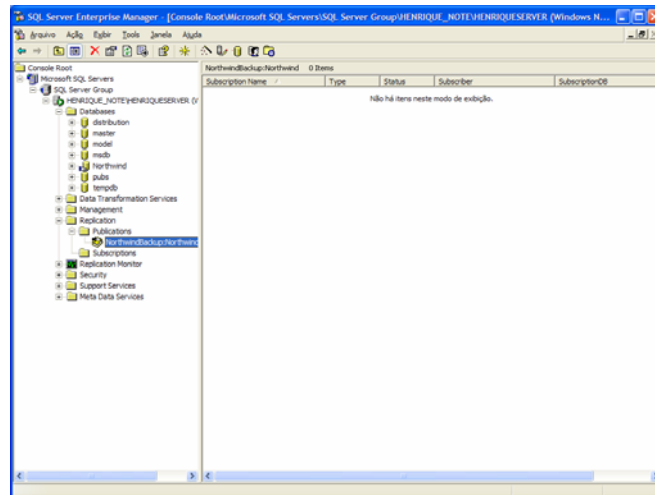


Criando Subscribers para a Publicação

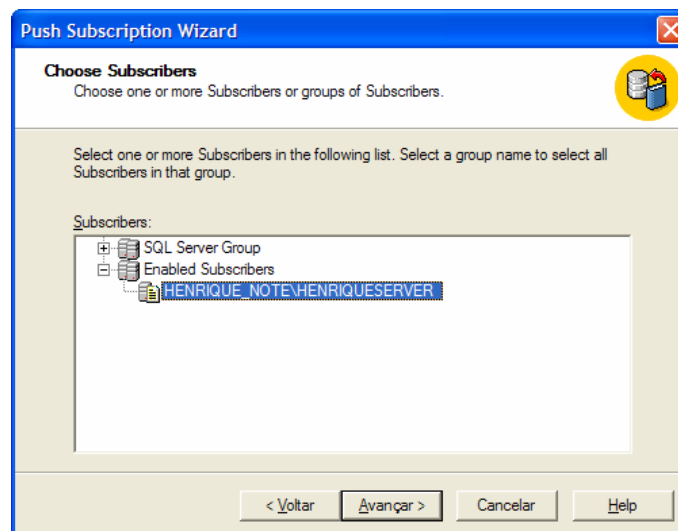
Após criarmos uma publicação para a força de vendas, vamos implementar a criação dos subscribers, que irão utilizar os dados que serão obtidos no processo de replicação.

Vamos criar uma subscrição para um novo banco de dados no servidor, que simulará um servidor remoto.

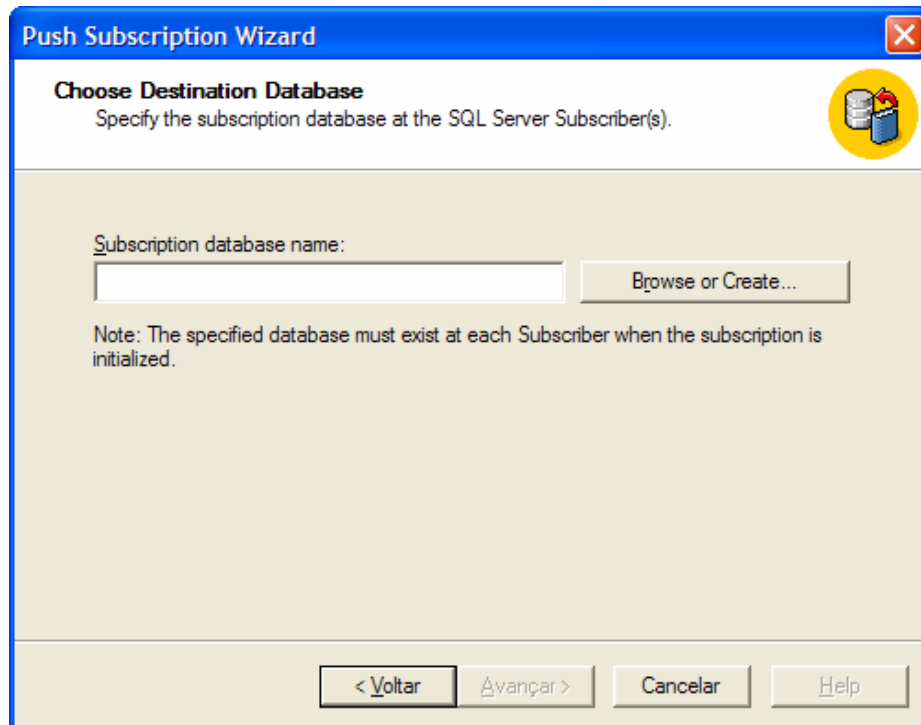
- Inicie o “Push Subscription Wizard”, e selecione o checkbox para visualizar as opções avançadas - Com o FieldSales: clique com o botão direito e selecione “Push New Subscriptions”



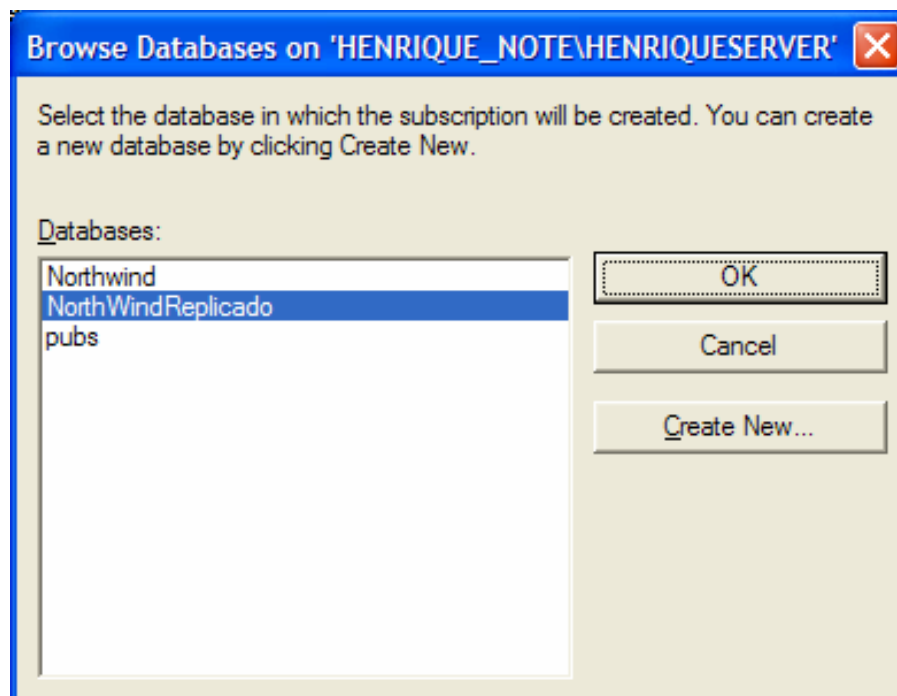
- Selecione o seu próprio servidor como sendo o Subscriber



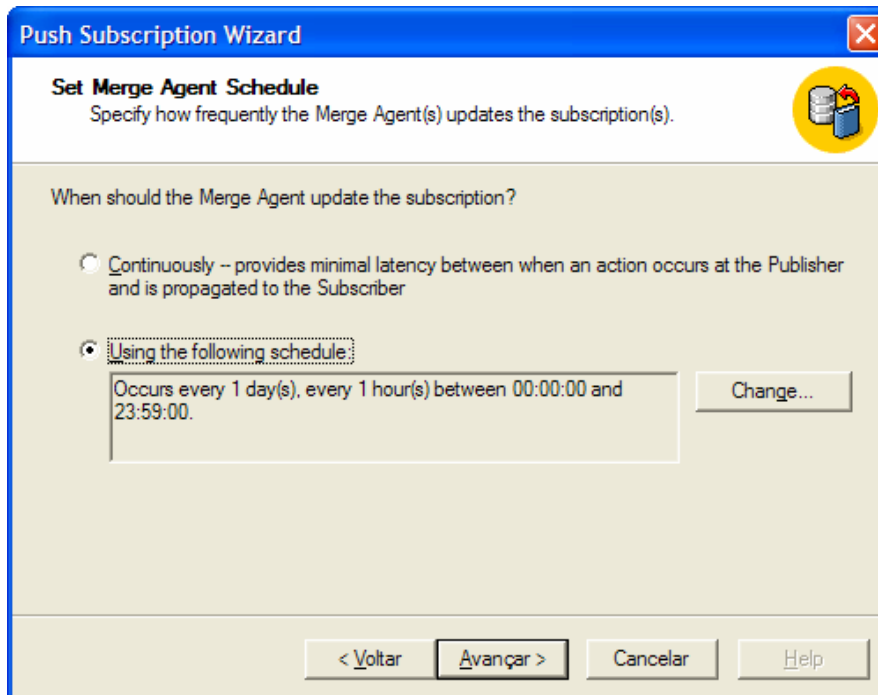
- Como database destino, crie um novo banco de dados chamado “SalesDavólio” e selecione o mesmo como destino



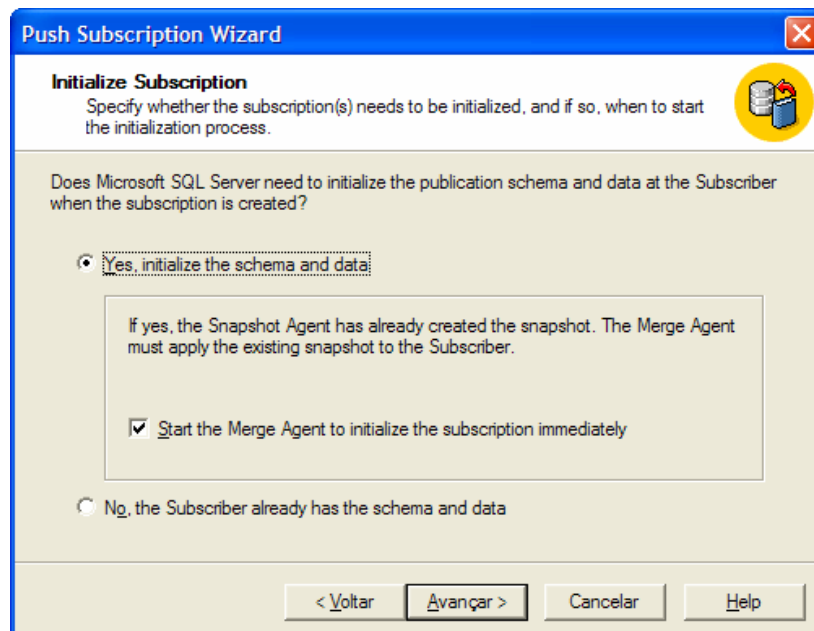
- Selecione o novo database



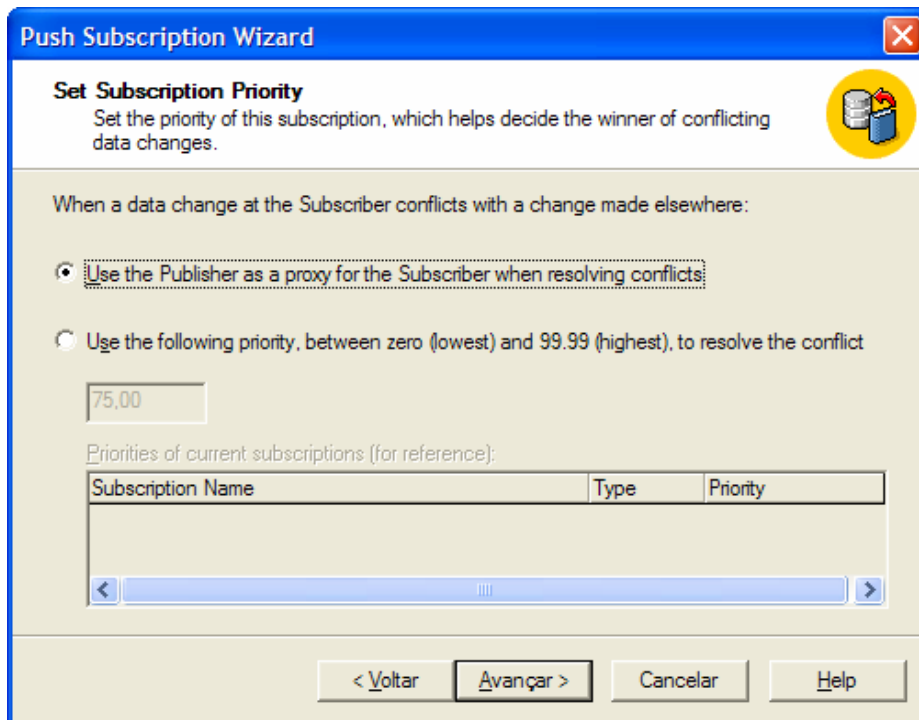
- Aceite as opções default para o agendamento e sincronização da Replicação



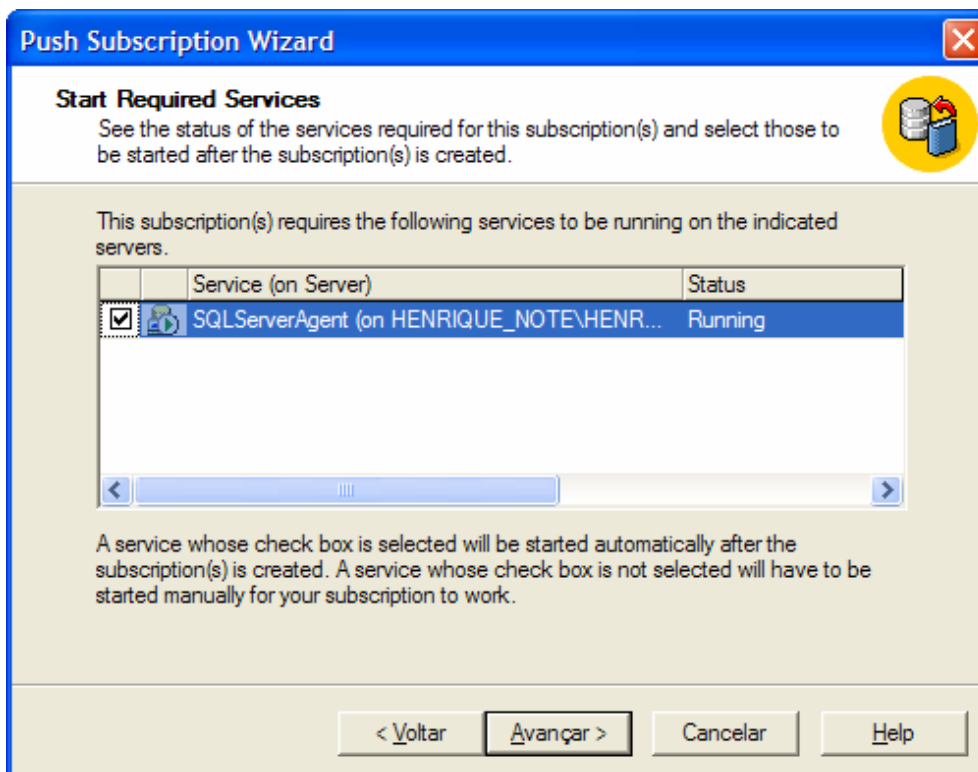
- Selecione “YES” para inicializar o schema e os dados no subscriber, então selecione o checkbox para iniciar o snapshot agent imediatamente.



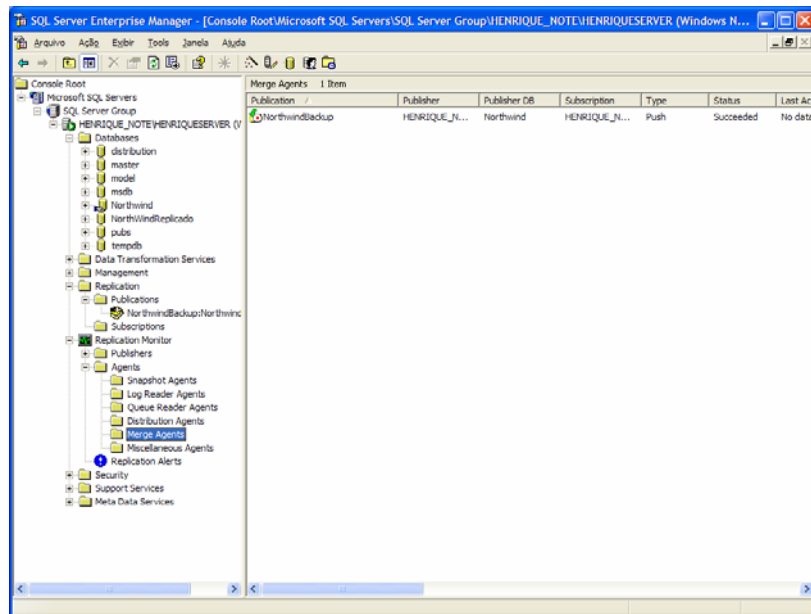
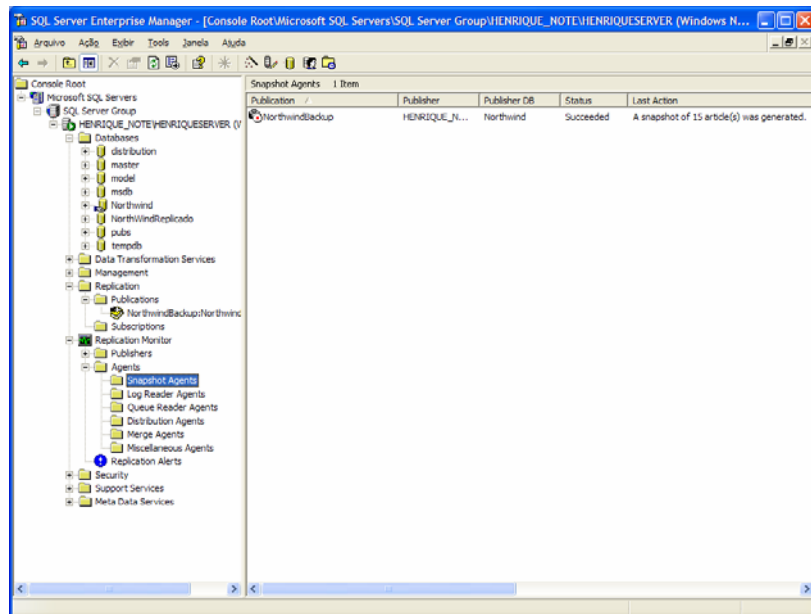
- Aceite as configurações padrão de prioridade na resolução de conflitos



- Certifique-se de que os serviço Agent estão no ar



- Com a configuração finalizada, verifique se o Snapshot Agent executou corretamente e inicie o Merge Agent manualmente para subscrever o database



- Verifique que o Subscriber recebeu os dados, use o Replication Monitor, SQL Server Agent ou Database information no Taskpad do Enterprise Manager