Pecuária e Abastecimento

Relatório Técnico 12

ISSN 1517-0330
Dezembro, 2001

Migrando do Paradox para o Interbase: a Experiência do AINFO

Embrapa

ISSN 1517-0330
Dezembro, 2001

Empresa Brasileira de Pesquisa Agropecuária
Centro Nacional de Pesquisa Tecnológica em Informática para a Agricultura
Ministério da Agricultura, Pecuária e Abastecimento

Relatório Técnico 12

Migrando do Parodox para o Interbase: a Experiência do AINFO

João Francisco Gonçalves Antunes
Carlos Henrique Cantu

Campinas, SP
2001

Embrapa Informática Agropecuária
Área de Comunicação e Negócios (ACN)
Av. Dr. André Tosello s/nº
Cidade Universitária "Zeferino Vaz" – Barão Geraldo
Caixa Postal 6041
13083-970 – Campinas, SP
Telefone/Fax: (19) 3789-5743
URL: http://www.cnptia.embrapa.br
Email: sac@cnptia.embrapa.br

Supervisor editorial: *Ivanilde Dispato*
Normalização bibliográfica: *Marcia Izabel Fugisawa Souza*
Capa: *Intermídia Publicações Científicas*
Editoração eletrônica: *Intermídia Publicações Científicas*

1ª edição

Antunes, João Francisco Gonçalves.

Migrando do Paradox para o Interbase: a experiência do AINFO / João Francisco Gonçalves Antunes, Carlos Henrique Cantu. — Campinas : Embrapa Informática Agropecuária, 2001.

25 p. : il. — (Relatório técnico / Embrapa Informática Agropecuária ; 12)

ISSN 1517-0330

1. Sistema gerenciador de banco de dados. 2. Interbase. 3. Automação de bibliotecas. 4. AINFO. 5. Recuperação de informação. I. Cantu, Carlos Henrique. II. Título. III. Série.

CDD – 005.7565 (21.ed.)

Sumário

Migrando do Paradox para o Interbase: a Experiência do Ainfo

João Francisco Gonçalves Antunes[1]
Carlos Henrique Cantu[2]

Resumo

O objetivo deste trabalho é relatar a experiência na migração do sistema de automação das bibliotecas da Embrapa – AINFO, em Paradox para o Interbase, versão gratuita com código aberto, disponibilizado pela Borland Software Corporation em setembro de 2000. O método utilizado foi o de estudar a viabilidade, performance e aplicação dos componentes de acesso nativo Interbase Express (IBX) e Interbase Objects (IBO) que dispensam a Borland Database Engine (BDE) para conexão do sistema ao banco de dados.

Termos para indexação: Interbase; Sistema gerenciador de banco de dados; Automação de bibliotecas; Recuperação de informação; AINFO.

---

[1] Bsc. em Matemática Aplicada e Estatística, Pesquisador da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo – 13083-970 – Campinas, SP. (joaof@cnptia.embrapa.br)

[2] Consultor da Embrapa Informática Agropecuária no desenvolvimento do AINFO 2001. (warmbooter@warmboot.com.br)

Migrating from Paradox to Interbase: the Experience of the Ainfo

Abstract

The objective of this paper is to present the database migration experience, regarding Embrapa´s software system for libraries automation – AINFO, from Paradox to Interbase, free version open source, available by Borland Software Corporation in September, 2000. The method employed encompassed studying the feasibility, performance and application of the native database access components Interbase Express (IBX) and Interbase Objects (IBO) which do not make use of Borland Database Engine (BDE) for connecting the system to the database.

Index terms: Interbase; Database managment system; Library automation; Information retrieval; AINFO.

Introdução

Os avanços tecnológicos atuais estão obrigando as empresas, produtos e serviços a mudanças e adaptações em velocidade sem precedentes. Conforme Oliveira (1993), o ritmo acelerado de mudança está repercutindo até nos segmentos empresariais mais conservadores ou que nunca necessitaram de suporte tecnológico. Dessa forma, a tecnologia da informação começa a alterar a natureza da administração e a afetar de maneira contundente, o direcionamento e o ritmo das mudanças.

Uma biblioteca hoje adquire o papel de uma consultora de informação, auxiliando seus usuários a localizarem os assuntos solicitados, utilizando os recursos da computação (Shiozawa, 1993). Através desses recursos computacionais, usuários de diversas localidades podem acessar remotamente os acervos de bibliotecas espalhadas pelo mundo, além de bases de dados públicas.

É dentro desse contexto que a informática tem exercido influência fundamental no funcionamento de bibliotecas e serviços de informação. Para gerenciar as bibliotecas com eficiência, os sistemas de recuperação de informações armazenam e proporcionam rápido acesso, através do gerenciamento de informações textuais, CD-ROMs e acesso on-line a bases de dados externas.

Segundo Rowley (1994), a maioria das bibliotecas lidam tanto com o gerenciamento de seus serviços quanto com a recuperação de informações. As aplicações mais recentes de informática em bibliotecas e serviços de informação consideram essas duas áreas como um campo integrado.

Atualmente, os sistemas de computação voltados para automação de bibliotecas como o AINFO da Embrapa, de acordo com Lenk (2000), concentram-se nas atividades de processamento de informações, aquisição de materiais, catalogação, controle de empréstimo, manutenção do acervo bibliográfico, controle de periódicos, empréstimos entre bibliotecas, além de outras atividades relevantes, como é o caso de oferecer a informação para acesso comunitário.

Desde sua primeira versão, em 1991, o AINFO tem evoluído para incorporar novas funcionalidades. Inicialmente foi desenvolvido nos ambientes MS-DOS e Unix mas com a melhoria do parque computacional da

Embrapa foi necessário e possível evoluí-lo para uma plataforma de software mais atual e com interface gráfica mais amigável, originando o AINFO em Paradox, baseado num sistema gerenciador de banco de dados em ambiente Windows. Porém, o sistema começou a apresentar problemas de perda de dados e lentidão com o acesso multiusuário em rede.

O problema principal a ser solucionado é o de modernizar a infra-estrutura de software utilizada para organização, gerenciamento e disseminação de informação técnico-científica. Para que o AINFO não se torne uma ferramenta obsoleta e desvinculada das necessidades dos usuários, deixando de atender as demandas dos diferentes segmentos do negócio agrícola, é necessário evoluí-lo. Isso deve ser feito em decorrência do surgimento de novas ferramentas de software, no caso o Interbase e seus componentes de acesso nativo, com o objetivo principal de sanar os problemas surgidos com a utilização do Paradox.

Este trabalho relata a experiência na migração do AINFO em Paradox para o Interbase, através do estudo de viabilidade e aplicação dessas novas tecnologias atualmente disponíveis.

Material e Métodos

Como forma de viabilizar soluções de informática para a incorporação de novos componentes tecnológicos e novas funcionalidades demandadas pelos usuários do AINFO, visando torná-lo um instrumento cada vez mais ágil e eficiente para a armazenagem, organização, recuperação e disseminação de informação, foi realizado um estudo exploratório das necessidades e situação atual do sistema, levando em consideração o aumento de sua flexibilidade e sua capacidade de processamento.

A partir desse estudo na utilização do AINFO em Paradox após a sua instalação na biblioteca da Embrapa Informática Agropecuária em janeiro de 2000, observou-se que o sistema apresentava problemas de robustez do banco de dados por causa da corrupção e perda de dados e, também, baixa performance em rede.

Com o advento da disponibilização gratuita do Interbase com código aberto pela Borland Software Corporation em setembro de 2000, decidiu-se

pela migração para esse verdadeiro sistema gerenciador de banco de dados (SGBD) (Silberschatz et al., 1999), utilizando componentes de acesso nativo que dispensam a Borland Database Engine (BDE) para conexão do sistema ao banco de dados.

Interbase

Para muitos, o nome Interbase é totalmente desconhecido, para outros é sinônimo de eficiência e fácil manutenção. O Interbase não é um produto novo pois está no mercado há mais de 10 anos e, nesse tempo, adquiriu respeito e admiração de muitos programadores, desenvolvedores e clientes, dentre eles podemos citar a NASA, Nokia, Boeing e o exército americano.

O fato de não ser tão reconhecido como o Oracle, o Microsoft SQL Server e outros servidores SQL, deve-se à falta de marketing e divulgação adequada por parte da Borland nos meios especializados. No entanto, essa imagem tende a mudar rapidamente, pois agora o Interbase passa a ter seu código distribuído livremente pela Internet com licenças de utilização gratuitas. Isso quer dizer que não é mais necessário utilizar as ultrapassadas base de dados Paradox, tendo um banco de dados poderoso, eficiente e seguro sem ter custo adicional por isso (Cantu, 2001).

A seguir são citadas algumas características do Interbase:

- Sistema de multiversões

Uma das maiores vantagens do Interbase sobre seus concorrentes é o uso de um sistema otimizado de concorrência no acesso ao banco de dados. Seguindo uma linha totalmente diferente da maioria dos bancos de dados, onde há o bloqueio de páginas inteiras, o Interbase trabalha com várias versões dos registros mantendo assim uma visão consistente dos dados durante uma transação, independente de alguma informação ter sido alterada após a transação ter sido iniciada (*versioning*).

Para exemplificar, existem situações onde várias pessoas estão utilizando um banco de dados ao mesmo tempo, gerando relatórios, atualizando dados, etc. Imagine que entre a geração de um relatório na tela e a sua impressão definitiva, um outro usuário atualize uma informação. Nos bancos de dados convencionais, o relatório impresso seria diferente do gerado na tela porque os dados alterados interferiram no resultado final

do relatório. No Interbase, através dos diversos tipos de isolamento transacional é possível ter a mesma imagem dos dados inalterada pelo tempo que for necessário, sem impedir que outros usuários continuem acessando o banco.

Quando um usuário altera uma informação de um campo em um registro, o Interbase cria um novo registro com os campos que tiveram seus dados alterados. Esse registro contém um *timestamp* (marcador de tempo) que permite ao banco de dados saber qual é a informação mais atualizada. O registro contendo os dados anteriores não é descartado enquanto houver uma transação ativa fazendo uso dele. Após o registro ser liberado, ele é automaticamente marcado para exclusão, excluído ou tem seu espaço reutilizado pelo banco de dados. Esse mecanismo também é utilizado para saber se mais de um usuário tentou alterar os mesmos dados num determinado período, o que gera automaticamente um evento de erro no banco de dados (*deadlock*) garantindo assim a consistência das informações.

- Arquitetura

Possui uma arquitetura multiprocessamento compartilhada que aumenta a performance e otimiza o uso de recursos de sistemas, especialmente por um grande número de usuários. Isso permite ter vários clientes num mesmo servidor mantendo alta velocidade de reposta.

- Multiplataforma

Atualmente está disponível para os sistemas operacionais Win9x, NT, 2000, Solaris e foi o primeiro banco de dados profissional a ter uma versão para o Linux.

- Configuração e manutenção

Uma outra grande vantagem do Interbase é quanto à definição das características físicas do banco de dados. Em muitos servidores SQL, o administrador do banco de dados (DBA) deve previamente estimar, definir e preparar um espaço do disco rígido para ser usado pelo banco. Os bancos de dados criados no Interbase são arquivos comuns do sistema operacional que crescem e diminuem conforme a necessidade, sem que seja necessário a intervenção do DBA. A manutenção e configuração de um banco de dados Interbase é praticamente zero.

Mantém sua base de dados limpa e consistente através de rotinas automáticas de manutenção executadas geralmente quando o banco de dados está em estado de espera (*idle*). Cada banco de dados Interbase consiste de pelo menos 1 arquivo (com extensão .GDB), pois pode ter vários, onde todas as informações, tabelas, índices, etc. ficam armazenados, facilitando as operações de administração e backup.

- Suporte a domínios

Suporta o uso de domínios na definição de campos. Na verdade, todo campo criado em uma tabela no Interbase possui um domínio próprio, criado automaticamente pelo sistema ou definido previamente pelo DBA. Com essa tecnologia, fica muito fácil fazer alterações em cascata em campos do mesmo tipo. A característica do Interbase é, que quando um domínio é alterado, os dados armazenados utilizando esse domínio não são convertidos imediatamente para o novo formato, o que ocasionaria uma certa sobrecarga (*overhead)* no servidor. Os dados que já estão gravados somente serão convertidos quando forem utilizados mas isso fica transparente para o usuário. A execução de um backup seguido da sua restauração também faz com que todos os campos sejam convertidos para o novo formato.

- Otimização de consultas (*queries*)

O próprio servidor otimiza uma consulta de maneira a buscar os resultados da maneira mais eficiente e rápida, porém também é possível definir um plano de otimização manualmente.

- Campos BLOB

Suporte a campos BLOB que podem armazenar qualquer tipo de dado desde textos, imagens, gráficos, som e binários.

- Protocolos

Suporte a diversos protocolos: Local, TCP/IP, NetBeui, IPX/SPX (Novell).

- Funções definidas pelo usuário (UDF)

Esse é um recurso muito poderoso do Interbase, onde é possível utilizar dentro do próprio banco de dados funções criadas pelo usuário usando qualquer linguagem que gere uma biblioteca dinâmica. UDFs também podem ser usadas para executar aplicações externas ao banco de dados.

- Multitransacional

Suporte a múltiplas transações.

- Acesso nativo

Existem componentes de acesso nativo ao Interbase disponíveis no mercado tais como o Interbase Express (IBX) e o Interbase Objects (IBO).

- Campos de 64 bits

Suporte a campos numéricos inteiros de 64 bits utilizando uma técnica em que números de ponto flutuante podem ser armazenados no banco de dados como inteiros, removendo assim o risco de erros de arredondamento. Toda a conversão entre inteiros e decimais é transparente ao usuário.

- Vetores multidimensionais (*arrays*)

Suporta vetores multidimensionais com até 16 dimensões muito usados em aplicações financeiras e científicas, sendo possível armazená-los em um único campo no banco de dados, o que simplifica o projeto da aplicação e aumenta a performance.

- Banco de dados distribuídos

É possível trabalhar com o Interbase num ambiente de banco de dados distribuído por ser um verdadeiro servidor de bancos de dados distribuídos SQL, que permite que cada consulta do sistema de banco de dados retorne a informação para qualquer outro servidor.

O Interbase gerencia transações com servidores múltiplos de maneira rápida porque o processamento é feito com *commit* em 2 fases, o que assegura que as atualizações sejam feitas sem intervenção da aplicação. Toda vez que uma transação abrange 2 ou mais servidores, o Interbase primeiro investiga os servidores participantes para assegurar de que eles estejam em atividade e rodando, enviando então o comando *commit* para completar a transação.

Por isso, também provê recuperação distribuída para um *commit* em duas fases, assegurando a recuperação completa sem o risco de um único ponto de falha, pois a coordenação da submissão é distribuída entre todos os servidores, reduzindo assim a necessidade de administração dos dados.

No evento em que a transação não possa ser submetida em todos os servidores, um *rollback* da transação inteira é automaticamente realizado em todos os servidores.

- Replicação

O Interbase oferece suporte à replicação e sincronização entre múltiplos bancos de dados através de componentes de terceiros que até o momento não são gratuitos, mas de custo bem acessível.

Borland Database Engine (BDE)

A origem da BDE vem desde antes da criação das ferramentas de desenvolvimento, antes mesmo da versão 16 bits do Windows. Ela apareceu como a máquina por trás do Paradox 3.5 e do Quattro Pro e foi modificada logo depois para fornecer uma interface de programação (API) para o DBaseIII-Plus, FoxBase, bem como uma API para os drivers *SQLLinks* para diversos servidores SQL. Ela migrou para a plataforma Windows na primeira versão imatura do Paradox e um pouco depois, na versão 3.x, recuperou sua boa reputação dos tempos de MS-DOS, com uma versão rápida e estável que acompanhava a versão 16 bits do Paradox 5.

A BDE continuou a crescer como uma camada de API entre plataformas de desenvolvimentos e *drivers* abertos para conexão às diversas plataformas de banco de dados. Apesar de ter seus defeitos, ela foi responsável por parte do sucesso do Delphi (Inprise Corporation, 1999) nos últimos 5 anos.

A seguir, são listadas algumas considerações do porque não utilizar a BDE:

- Tamanho

Desenvolver aplicações de banco de dados que precisam da BDE aumenta em muito o tamanho da instalação. Restrições de licença requerem que ela seja distribuída sem modificações, com total suporte ao Paradox. Essas restrições existem em parte porque a BDE utiliza tabelas desse padrão para fazer *cache* e para outros propósitos, mesmo que o Paradox não seja o banco de dados utilizado pela aplicação.

- Fragilidade e custo para desenvolvimento voltado à Internet

Outros problemas de distribuição ocorrem quando a BDE é usada em aplicações Web. O suporte a serviços de sistema desse tipo tem um alto custo. A fragilidade da BDE, devido, às vezes, à negligência do instalador em fazer uma configuração correta, com seu tamanho exagerado, pode prejudicar as facilidades oferecidas pelo Interbase como um banco de dados de configuração e manutenção baixíssima.

- Nivelando por baixo

Desconsiderando os dados de economia, a maior falha da BDE é o fato de não utilizar por completo toda a capacidade dos diversos bancos de dados. O propósito da BDE é de esconder as diferenças entre os vários recursos dos bancos e, por compatibilidade com alguns que não suportam múltiplas transações ou *commits* em duas fases, acaba forçando as aplicações a utilizar uma transação por conexão mesmo que o banco de dados utilizado suporte esses recursos. Isso acaba com a capacidade de processamento assíncrono e conexões a múltiplos servidores.

Em aplicações Delphi, ela prejudica até mesmo o Paradox oferecendo um conjunto limitado de comandos SQL (Date, 1989) que sacrifica a velocidade da programação nativa. Em geral, no entanto, o suporte da BDE aos banco de dados nativos é razoavelmente aceitável para aplicações simples de acesso local.

- Performance

Diversos fatores sobre uma implementação usando a BDE inibem o desenvolvedor de utilizar o Interbase com toda a sua performance e eficiência. A combinação da BDE mais *driver* implica em 2 camadas de código interfaceável que diminui a velocidade da conexão entre o aplicativo e a API do Interbase, sem falar da falta de suporte a multitransações e multiservidores.

O fato dos componentes visuais (VCL) padrão *TTable* e *TQuery* servirem a um cenário genérico de banco de dados definiu ainda mais limites, inibindo a escalabilidade e a performance. Em várias circunstâncias, esses limites começam a tornar as operações lentas e o desenvolvedor sente necessidade de uma interface direta que utilize todas as vantagens do banco de dados.

- Final da vida

Agora que o Delphi começou a ser portado para outros sistemas operacionais como o Linux, o chamado Kylix, a Borland já produziu uma nova camada de conectividade chamada DBExpress que utiliza os objetos de acesso a dados dbCLX para conexão ao banco de dados.

Tudo indica que a Borland produziu a última versão da BDE e por isso uma nova solução de conectividade para Windows independente da BDE já é necessária.

Comparação entre o IBO e o IBX

- Arquitetura e peculiaridades

O IBX possui inúmeros comportamentos peculiares com sua estrutura de buffer primitiva. Por isso, muitos desenvolvedores costumam usar o IBX somente para processos simples e de baixo nível que não requerem a intervenção dos usuários finais, usando uma fina camada para enviar e receber as instruções SQL entre o servidor e seus clientes. A não ser que a aplicação esteja fazendo algo desse tipo, o melhor é não depender do IBX.

O IBO foi cuidadosamente desenhado para fazer todo o trabalho necessário para evitar peculiaridades, de maneira que as aplicações se comportem como as pessoas esperam. O gerenciamento de buffer do IBO é muito flexível e contém SQL *smarts* que fazem com que muitas tarefas possam ser executadas no servidor ao invés de trazer os registros e processá-los no cliente. Também permite que a combinação de parâmetros de entrada, filtros, métodos trabalhem em harmonia sem que haja necessidade de desabilitar os SQL *smarts*, continuando o processamento no servidor sem perder nenhum recurso disponível no *dataset*. O IBO é o resumo das arquiteturas de acesso a dados cliente/servidor (Wharton, 2001).

- Tratamento de transações

O tratamento das transações do IBO é muito melhor do que no IBX porque não se limita apenas ao nível físico. Isso é um grande ponto de confusão para as pessoas que estão aprendendo o IBX e que já tem experiência passada com a BDE, devido à abstração que a BDE faz em uma unidade lógica, tratando a complexidade restante no nível físico. Isso possibilitou uma maneira simples de criar aplicativos, mas infelizmente,

a BDE também introduziu alguns comportamentos indesejáveis na sua tentativa de tornar as coisas simples. No entanto, o IBO oferece todos os benefícios que a BDE oferece e ao mesmo tempo eliminou essas peculiaridades, possibilitando ter total controle das transações também no nível físico.

Como exemplo de como isso pode afetar suas aplicações, considere o seguinte cenário: no Interbase, uma transação física não deve ficar aberta por muito tempo. Sendo assim, é importante que se certifique que as transações sejam fechadas corretamente. Com o IBX, uma transação finalizada força que todas os *datasets* ligados a essa transação sejam fechados também. Nesse caso, a única opção é forçar o fechamento dos *datasets* periodicamente ou forçar uma desconexão no servidor caso algum usuário deixe um *dataset* aberto indefinidamente. Não há nenhuma sofisticação no IBX para tratar esse problema de uma maneira melhor.

O IBX também exige que uma transação seja iniciada antes que qualquer *dataset* seja aberto. Assim, as transações não são mais puramente para o processamento lógico de unidades de trabalho, mas elas também influenciam na leitura dos dados de alguma forma. Esse é o modo que o Interbase trabalha fisicamente, o que é muito complicado para a maioria dos propósitos de desenvolvimento de uma aplicação. Todo o seu código tem que ser escrito de maneira que todas as *datasets* sejam abertas e fechadas em harmonia com as transações, ao invés de voltar o foco em como as alterações são enviadas ao servidor e gerenciadas. Isso se torna complicado quando se tenta criar uma aplicação que seja adequada para a interação com o usuário final.

No IBO existe um nível bem inteligente acima da API que torna as transações muito mais amigáveis. Existe um nível físico que é a API, há o nível lógico que é onde as unidades de trabalho são definidas e, por último, há o nível de controle explícito de transações que depende totalmente do desenvolvedor. Há muitas sobreposições entre estes 3 aspectos e o IBO oferece controle total sobre eles. O desenvolvedor pode utilizar mecanismos explícitos para controle ou reagir ao estado lógico em que a transação está, dependendo do que o usuário executou.

No nível físico, é possível criar mecanismos para garantir que uma transação não fique aberta por muito tempo. Vários estágios de controles de

timeout são ativados e gerenciados automaticamente, liberando a transação física para que os recursos do servidor sejam conservados e para que a transação mais antiga seja tratada cuidadosamente. Se alguma coisa estiver travando esse mecanismo, existem inúmeras propriedades disponíveis para descobrir o que está acontecendo e assim responder corretamente a isso. É realmente fácil configurar mecanismos para interagir com o usuário para resolver longas transações.

No IBO, é possível atribuir a cada *dataset* um comportamento específico quando uma transação é finalizada. Existem inúmeros comportamentos para serem escolhidos como, por exemplo, simplesmente trazer todos os registros e armazená-los em memória ou atualizá-los automaticamente, de maneira que sejam recuperadas todas as alterações visíveis para a nova transação. Se todos os registros não forem recuperados, pode-se invalidar o cursor que faz com que a transação termine mantendo o *dataset* intacto. Na próxima vez que a transação for ativada, o cursor inválido é reconhecido e automaticamente um novo cursor é aberto, mantendo-se na mesma posição onde estava antes, com o mínimo de consumo do servidor.

Também pode-se fechar o *dataset*, ficando em modo de consulta. Isso é muito mais flexível que o método da BDE que sempre recupera todos os registros de um *dataset* aberto quando um *commit* ou um *rollback* é efetuado.

- Performance

Conforme relatos da experiência de vários usuários, tanto o IBO quanto o IBX mostram um ganho de performance entre 200% a 500% com relação à BDE, dependendo da operação, tais como inserções, edições, navegação e localização. Como o IBO utiliza um interpretador (*parser*) inteligente do lado do cliente para otimizar o diálogo com o servidor, cabe ressaltar que o aumento da performance é muito grande especialmente quando as conexões são lentas.

- Convertendo da BDE

A conversão de aplicativos que utilizam a BDE para a versão IBO é muito mais simples do que para a versão IBX porque o IBO suporta a maioria dos recursos da BDE. O IBX não suporta muitos desses recursos e tem um método significativamente diferente de implementar outros métodos,

que faz com que o desenvolvedor tenha um grande trabalho de conversão e de testes, antes de considerar o trabalho de conversão terminado. Os componentes do IBX são totalmente diferentes dos *datasets* baseadas na BDE, necessitando de revisões que afetam a interface dos componentes.

- Custo

O IBX acompanha a versão profissional do Delphi e IBO, por ser um componente de terceiros, pode custar de U$395,00 a U$ 595,00, dependendo dos módulos adquiridos. Mas, em contrapartida, oferece uma licença chamada *Trustware*, fornecendo o componente gratuitamente para quem tenha interesse em contribuir no desenvolvimento, documentação, exemplos, etc., desenvolvedores ligados a projetos *Open Source*, organizações sem fins lucrativos e também empresários que ainda não estejam com o seu produto no estágio de comercialização.

Resultados e Discussão

A partir do estudo de viabilidade, performance e aplicação das tecnologias de software hoje existentes, decidiu-se usar o IBO, que mostrou ser o pacote de componentes de acesso nativo ao Interbase mais profissional disponível na atualidade, com recursos e performance únicos. Oferece componentes que foram desenvolvidos desde o início para serem utilizados com o Interbase, com base na filosofia da arquitetura cliente/servidor, aproveitando ao máximo os recursos do banco de dados e que realizem operações otimizadas nesse tipo de arquitetura. Mantém compatibilidade com os componentes do Delphi e de terceiros, oferecendo muito mais recursos com excelente performance porque mesmo os componentes que descendem do *TDataset* utilizam internamente um componente nativo (*TIB_Query*) para realizar as operações de acesso a dados.

Com essa definição, iniciou-se o trabalho efetivo de migração do Paradox para o Interbase 6.01 e o IBO 4.2.Fn com a licença *Trustware* fornecida pelo fabricante à Embrapa. A seguir são detalhados os pontos-chave a serem considerados durante o processo de migração:

- Tipos de dados

O Interbase suporta basicamente todos os tipos de dados do Paradox. Apenas deve-se tomar cuidado na conversão das tabelas para o Interbase quando os mesmos apresentarem campos numéricos. Utilitários como o Datapump que acompanha o Delphi convertem os campos numéricos para *SMALLINT* que facilmente podem ter seus limites ultrapassados. A melhor opção é utilizar um campo inteiro no caso de campos do Paradox numéricos (sem casas decimais) e campos *NUMERIC* para os campos do Paradox que contenham casas decimais.

- Campos do tipo lógico

O Interbase não tem suporte a esse tipo de dados mas é perfeitamente possível substituí-lo por campos *CHAR* de 1 caracter, usando "V" para verdadeiro e "F" para falso ou, então, inteiros utilizando 0 para falso e 1 para verdadeiro.

- Campos do tipo data

No Interbase, é possível definir um campo somente para armazenar data e outro para armazenar hora, mas para isso o banco de dados deve utilizar o *Dialeto 3*, caso contrário data e hora são armazenadas no mesmo campo tipo *TIMESTAMP*.

Como o Interbase procura cada vez mais ser compatível com os padrões definidos para a SQL, o conceito de dialetos permite utilizar novos recursos que não são compatíveis com versões anteriores.

- *Triggers* e *Stored Procedures*

São funções e procedimentos que executam regras de negócio da aplicação no servidor o que aumenta tanto a performance quanto a segurança de que as operações sejam bem sucedidas, com a melhor eficiência possível.

- Campos auto-incrementais

Esse tipo de campo é muito utilizado no Paradox e pode ser simulado no Interbase através de *generators* e *triggers*, já que não é suportado nativamente.

Um *generator* é um contador interno do banco de dados que tem seus incrementos (positivos ou negativos) controlados através de instruções SQL disparados por *triggers* geralmente para atribuição das chaves primárias das tabelas. Os *generators* não ficam isolados por transações,

portanto, se for incrementado durante uma transação, não terá seu valor restaurado caso a transação não se complete e todas as outras transações em andamento receberão o valor atual.

Para simular um campo auto-incremento deve-se criar um *trigger* para o evento *BEFORE_INSERT* que é disparada antes de um novo registro a ser incluído no banco de dados. Nesse *trigger*, se o campo em questão tem o valor nulo, incrementa-se o *generator* obtendo o seu valor e atribuindo-o ao campo, caso contrário, não há alteração.

- Campos alfanuméricos

O Interbase possibilita o armazenamento de dados alfanuméricos através de campos *CHAR* e *VARCHAR*. A diferença básica entre eles é que o campo *VARCHAR* recupera os dados descartando os espaços em branco que possam existir no final. Ambos os tipos gravam os valores em formato comprimido para diminuir o espaço utilizado no armazenamento.

- Índices

O Interbase, assim como outros servidores SQL, não suporta índices que contenham funções em suas expressões. Se não houver outra alternativa, uma possível solução é criar um campo chave na tabela para armazenar a informação que formará o índice. Essa informação pode ser manipulada em *triggers* através de funções nativas ou definidas pelo usuário (UDF) para que assumam o formato desejado.

Os índices são sensíveis a maiúsculas e minúsculas. Se for necessário fazer uma filtragem dos registros através de uma instrução SQL, deve-se utilizar a função *UPPER* para transformar os caracteres em maiúsculas ou definir a ordenação apropriada na definição dos campos no banco de dados. Para a língua portuguesa deve ser utilizada a *Collate* PT_PT ou PXW_INTL850 que também faz a diferenciação entre os caracteres acentuados.

- Integridade referencial

Esse é um dos pontos mais importantes que diferenciam o Interbase do Paradox. Através da integridade referencial, pode-se garantir o relacionamento entre campos de diversas tabelas, impedindo que seja apagado um registro que tenha relação com outros.

A única observação a fazer é o cuidado na utilização desse recurso porque para cada relacionamento é criado um índice para os campos envolvidos e, portanto, pode afetar a performance do banco de dados bem como o aumento do espaço de armazenamento das informações.

- Transações

Esse também é um outro grande diferencial em relação ao Paradox. Através de transações, é possível isolar operações no banco de dados, assegurando que, se por algum motivo durante a transação ocorra um evento de erro, os dados que já foram alterados em qualquer tabela sejam revertidos ao seu estado anterior à transação (*rollback*).

É um recurso extremamente útil e deve ser utilizado sempre que possível para que a integridade dos dados seja mantida corretamente. O Interbase oferece vários tipos de isolamento para as transações, ou seja, como os dados alterados durante uma transação podem ser vistos e compartilhados por outras transações:

- *Shared, write*: permite que qualquer transação com modo de acesso à escrita e nível de isolamento *concurrency* ou *read committed* atualize os dados, enquanto que outras transações com esses níveis de isolamento e modo de acesso de leitura possam ler os dados.
- *Shared, read*: permite que qualquer transação leia os dados e que qualquer transação no modo de escrita possa atualizar os dados. Esse é o modo mais liberal.
- *Protected, write*: previne que outras transações atualizem dados. Outras transações com o nível de isolamento de *concurrency* ou *read committed* podem ler os dados, mas somente essa transação pode atualizar.
- *Protected, read*: não permite que nenhuma transação atualize os dados, mas permite que todas as transações possam ler os dados.

- Acesso aos dados via componentes VCL *TQuery* ou *TTable*

Essa continua sendo uma das dúvidas mais frequentes quando se começa a utilizar um banco de dados cliente/servidor. Se a tabela a ser acessada contém poucos registros, poucos campos e estiver numa aplicação local, não há problema em usar um componente *TTable* para o acesso porque

o volume de dados movimentados não será significativo. Caso contrário, e sempre na arquitetura cliente/servidor, deve-se usar um componente *TQuery*.

A maneira como a instrução SQL para o componente *TQuery* é escrita também pode torná-la lenta. Deve-se limitar o volume de dados retornado utilizando-se uma condição através da cláusula *WHERE*, principalmente se o sistema permitir que o usuário navegue pelos registros. Isso evita desperdício de processamento e conseqüente perda de performance do sistema, principalmente em rede.

A *TQuery*, inclusive com *JOIN*, tem a propriedade de ser atualizável (*Live Dataset*) onde a definição manual dos comandos de atualização (*INSERT, APPEND, UPDATE, DELETE*) em instruções SQL é raramente necessária.

Se é necessário percorrer rapidamente grande volume de dados como, por exemplo, na emissão de relatórios, a utilização de um cursor (*TIB_Cursor*) unidirecional é o mais recomendável, como no caso do AINFO.

- Travamento pessismista

Em tabelas com grande quantidade de campos *MEMO*, ou seja, que envolvem muita digitação de informação textual, caraterística do AINFO, pode ser utilizado um recurso do IBO que simula um travamento pessimista no Interbase, impedindo que um usuário consiga alterar um registro que já esteja sendo alterado por outra pessoa. Essa forma de implementação garante que as atualizações de um registro sejam gravadas sem o risco de receber um erro de *deadlock* e conseqüentemente sem perder todo o trabalho de digitação.

- Visões (*views*)

Uma *view* é uma tabela virtual que não é armazenada fisicamente no banco dados. Pode conter dados de uma ou mais tabelas ou de outras *views* e é usada para armazenar resultados de consultas freqüentes ao banco de dados. Também pode fornecer meios de segurança limitando o acesso de usuários a um subconjunto de dados, enquanto esconde outros dados relacionados.

No AINFO, foram criadas *views* que retornam os dados para recuperação de informação indexados em tempo-real. Isso garantiu um ganho de

performance, porque as *views* já existem no servidor, evitando uma sobrecarga caso o mesmo processamento fosse executado através de *queries* com múltiplos *JOINs* no cliente.

- Backup/*Restore*

As rotinas de backup e *restore* do banco de dados utilizam as novas funções da API do Interbase 6.01, tornando possível a integração de todo o processo dentro do próprio executável.

Conclusões e Comentários

Quando se entra no mundo cliente/servidor, a primeira coisa a fazer é selecionar um SGBD. As arquiteturas dos SGBDs variam amplamente e, como resultado, o comportamento deles em uma determinada situação também. Isso significa que, para selecionar o SGBD correto para uma aplicação, é preciso saber como os dados serão acessados e modificados e como o servidor se comportará em cada acesso aos dados. Isso requer uma compreensão clara dos tipos de transações, isolamentos e atualizações.

O sistema de versões do Interbase tem uma vantagem clara porque pode processar transações de leitura e escrita concorrentemente e ainda provê o isolamento serializável para garantir segurança. Esse sistema também provê uma rápida recuperação depois de falhas pois não há arquivos de *Log* a serem processados. Além disso, o Interbase tem um tamanho pequeno, necessita de menos memória, é auto-configurável, altamente escalável porque está disponível para vários sistemas operacionais e ainda possui recursos que competem de igual para igual com outros SGBDs mais conhecidos, com a vantagem de ser gratuito e de código aberto.

Para os usuários que estão migrando sistemas do Paradox para o Interbase, o IBO emula praticamente todos os recursos (propriedades, métodos, componentes) da BDE, fazendo com que a migração de um sistema já existente seja razoavelmente rápida. Nenhum outro pacote de componentes para o Interbase possui essas características, fazendo com que apenas alguns ajustes sejam necessários para que o sistema portado comece a operar com acesso nativo.

Além disso, como o IBO também oferece um ramo de componentes que são derivados da classe *TDataset*, mantém compatibilidade com todos os controles de acesso ao banco de dados do Delphi e de terceiros como, o *InfoPower, RxLib,* etc.

Após a Borland ter disponibilizado o código do Interbase 6 publicamente, a comunidade *Open Source* organizou-se e começou a desenvolver um projeto paralelo baseado no código do Interbase chamado FireBird. Desde sua criação, o FireBird já solucionou diversos problemas e vem prometendo novos recursos em sua versão 1.0, que está prestes à ser lançada. O FireBird mantém total compatibilidade com a versão atual do Interbase e o fabricante do IBO comprometeu-se a mantê-lo compatível com o FireBird, inclusive com as novas implementações que ele vier a ganhar no futuro.

A experiência aqui relatada culminou com um excelente resultado que foi a implementação do AINFO 2001. Além de servir como base para migração de outros sistemas em Paradox para o Interbase pode, também, orientar o desenvolvimento de novos sistemas que tenham esse tipo de arquitetura.

Referências Bibliográficas

CANTU, C. H. *Interbase-BR.* Disponível em: <http://www.warmboot.com.br/ib>. Acesso em: 10 dez. 2001.

DATE, C. J. *Guia para o padrão SQL*. Rio de Janeiro: Campos, 1989. 189 p.

INPRISE CORPORATION. *Borland Delphi 5*: Enterprise. Scotts Valley, 1999. CD-ROM.

LENK, L. M. *Organização, recuperação e disponibilização de informação técnico-científica em bibliotecas - desenvolvimento do AINFO*. Campinas: Embrapa Informática Agropecuária, 2000. (Embrapa. Programa 14 – Intercâmbio e Produção de Informação em Apoio às Ações de Pesquisa e Desenvolvimento. Projeto 14.2000.369). Projeto em andamento.

OLIVEIRA, D. de P. R. de. *Sistemas de informações gerenciais*: estratégicas, táticas, operacionais. 2. ed. São Paulo: Atlas, 1993. 274 p.

ROWLEY, J. *Informática para bibliotecas.* Brasília, DF: Briquet de Lemos/Livros, 1994. 307 p.

SHIOZAWA, R. S. C. *Qualidade no atendimento e tecnologia de informação*. São Paulo: Atlas, 1993. 132 p.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. *Sistema de banco de dados*. São Paulo: Makron Books, 1999. 778 p.

WHARTON, J. *IBObjects home page*. Disponível em: <http://www.ibobjects.com>. Acesso em: 10 dez. 2001.

Informática Agropecuária

MINISTÉRIO DA AGRICULTURA,
PECUÁRIA E ABASTECIMENTO