Alguns conceitos Básicos sobre o IBO (IBObjects)

Resumo: Esse artigo provê alguns conceitos básicos sobre vários componentes incluídos no pacote IBObjects. O objetivo é dar aos novos usuários uma visão introdutória das propriedades comuns a detalhes de configuração para que possam iniciar a usar os componentes.

Para um maior detalhe dos conceitos básicos e das técnicas, por favor verifique o manual "Getting Started with IB Objects", disponível através dos canais de distribuição do IBObjects.

Conectando os componentes de Acesso a Dados

Os componentes IBO de acesso a dados interagem e se comportam na maioria das vezes da mesma maneira que os componentes de acesso a dados básicos do Delphi. O modelo tradicional possui um banco de dados, um dataset e um datasource. Ocorre o mesmo no IBO. Todavia, com bancos SQL, como Interbase/FireBird existe um fator adicional que deve ser considerado, as Transações.

Até então toda a referência a transações vinha sendo encapsulada pelo componente TDatabase do BDE (Borland Database Engine). O BDE usa uma conexão e uma transação embutidas em uma única interface, porém o InterBase/FireBird não está limitado a uma única transação por conexão.

Pelo fato do Interbase/Firebird possuir a habilidade de ter transações múltiplas e simultâneas em cada conexão, o IBO foi projetado para implementar as transações em um componente separado, específico para isso. Desta maneira é possível ter um único componente para conexão e vários componentes de transação fazendo referência a ele simultaneamente.

Outra característica importante do Interbase/Firebird é que suporta que uma única transação utilize mais de um banco de dados (multi-database). Desta forma também foi necessário projetar o IBO para administrar o caso em que uma transação referencia múltiplas conexões concorrentemente. O IBO permite que você tire proveito dessa poderosa e avançada característica de maneira bastante fácil. Use as propriedades IB_Connection1 e IB_Connection2 do componente TIB_Transaction para especificar e incluir conexões adicionais em um ambiente de transações entre vários bancos. Se mais que três conexões são necessárias (Default, IB_Connection1 e IB_Connection2), utilize o método AddConnection() para adicionar quantas conexões forem necessárias.

Resumindo, no IBO um Dataset não referencia simplesmente um Database. Ele também tem que fazer referência a uma transação. É isso o que determina em qual contexto os dados são lidos ou escritos no servidor. Tanto comandos/declaração SQL quanto os datasets no IBO possuem as propriedades Database e Transaction. Elas são IB_Connection e TIB_Transaction, respectivamente.

Com o objetivo de termos as coisas mais flexíveis os seguintes comportamentos foram implementados nas declarações SQL e nos Datasets.

Conexão Padrão

Se não existir uma conexão pré-definida, seja em um Dataset ou em um declaração SQL, então será checada se na Sessão default existe uma conexão. Se existir, ela irá se linkar automaticamente as propriedades pertinentes essa conexão padrão pré-existente. Esse é o comportamento típico para o primeiro componente de conexão (TIB_Connection) criado na aplicação.

Se existir somente uma única conexão para a aplicação toda normalmente não é necessário prestar muita atenção a isto, uma vez que o IBO irá automaticamente configurar as coisas para você.

Transação Padrão

Se uma referência a uma transação não é implementada ou em uma declaração SQL ou através de um Dataset, então o IBO gera uma transação internamente, de maneira transparente e automática. Isso proporciona um nível de isolamento que permite que a transação sempre enxergue o estado "Commited" do banco de dados, realizando um "auto-commit" em quaisquer mudanças que forem postadas. Desta maneira, se você projetar uma aplicação que não utiliza uma camada de transação, ela irá funcionar do mesmo jeito. Nesta abordagem os Datasets se comportam como se não houvesse o conceito de transação.

Modo "Normal"

Normalmente uma declaração SQL ou uma Query apontam suas propriedades IB_Connection e IB_Transaction para os componentes apropriados. A propriedade IB_Connection pode ser apontada para um TIB_Connection, TIB_Database, TIBODatabase ou qualquer sub-classe derivada deles. A propriedade IB_Transaction aponta para o componente TIB_Transaction ou qualquer uma de suas sub-classes.

Emulando o BDE

O Componente TIBODatabase foi projetado para fazer o que o TDatabase da BDE faz. Ele combina uma conexão com uma transação interna e por padrão faz com que todos os Datasets e declarações SQL que estejam ligados a ele compartilhem a mesma transação. Mesmo com esse comportamento é possível pingar componentes TIBOTransaction adicionais e configura-los, se assim desejado. Nesta abordagem a emulação do BDE está presente e ao mesmo tempo pode-se adicionar explicitamente transações adicionais, tirando proveito dessa abordagem.

Nota: Os componentes TIB_Database e TIBODatabase trabalham muito bem quando a propriedade CachecUpdates é definida como seu modo principal de uso. Use o TIBODataset se estiver usando componentes baseados em TDataset e TDatasource, da VCL do Delphi.

Revisado para o IBO 4

Conectando a controles Data-Aware

O IBO permite o mesmo tipo de conectividade em seus controles que a encontrada nos controles standard da VCL do Delphi. Cada controle terá uma propriedade DataSource e uma DataField, se aplicável ao caso.

Pelo fato dos controles IBO não serem compatíveis aos controles standard da VCL do Delphi, foi necessário criar um componente TIB_Datasource que age quase que identicamente ao seu equivalente na VCL, o TDatasource. Ele faz referência ao TIB_Dataset que pode ser um TIB_Cursor ou um TIB_Query.

Se você ficar se perguntando o porquê disto, é que o IBO foi iniciado muito tempo depois do Delphi 3 e da classe virtual TDataset introduzida por este. Então a única forma que encontrei de conseguir criar a funcionalidade que eu necessitava em minhas aplicações e tirar todo o proveito do InterBase/ FireBird foi escrever minha própria camada de acesso a dados e também controles baseados nas APIs do InterBase e nas classes TComponent, TWinControl, TCustomEdit, etc.

Classes TIBO* para compatibilidade com a VCL

Desde a introdução da classe virtual TDataset pelo Delphi 3 eu escrevi os componentes TIBODatabase, TIBOTable, TIBOQuery e TIBOStoredProc para se adaptarem aos componentes de acesso da paleta Standard e assim prover compatibilidade para todos os controles da VCL. Isso foi possível escrevendo uma camada derivada do TDataset que faz a ligação com uma instância interna do TIB_Query.

Além de permitir a compatibilidade com a VCL, isso permite que o IBO seja utilizado com geradores de relatórios como Report Printer Pro, Quick Report e outros.

Os controle que acompanham o IBO não são compatíveis com os controles da paleta Standard da VCL. Para continuar a usar os controles da VCl e controles de terceiros (compatíveis com a VCL) você deve usar o TDatasource padrão do Delphi e os objetos de acesso a dados do IBO iniciados por TIBO*.

As Classes TIB_*

Controles nativos IBO somente podem ser usados com componentes de acesso a dados também nativos: TIB_DataSource, TIB_Cursor, TIB_Query and TIB_StoredProc.

As Classes TIB_xxxxxSource

Os componentes StatementSource, ConnectionSource and TransactionSource foram desenvolvidos com o objetivo de deixar formulários ou outros controles “fora” de declarações, conexões ou transações específicas, respectivamente. Você pode ver esses componentes em funcionamento olhando os controles TIB_StatementBar, TIB_ConnectionBar ou TIB_TransactionBar.

Eu freqüentemente desenvolvo formulários sem dependência de conexões ou transações. Em run-time eu faço uma linkagem rápida no contexto de transação e conexão simplesmente associando uma referência da transação e da conexão aos componentes ConnectionSource e TransactionSource presentes no formulário.

Nos eventos AfterAssignment eu tenho uma rotina que replica essa referência a todos os datasets no formulário.

Resumo

Suas alternativas são:

Itens	Opção A	Opção B
Datasource	TIB_Datasource	TDatasource
Componentes de acesso (Data-aware)	Nativos IBO, ou seja, todos iniciados por TIB_*	baseados no IBO TDataset (TIBOQuery e TIBOTable)
Controles visuais (Grids, Edits, etc.)	Nativos IBO	Padrão da VCL

Aplicações “Híbridas” ou Mistas

Para os indecisos: Sinta-se a vontade para fazer uma mistura de componentes como TIBO* + controles VCL e TIB_* + controles nativos do IBO na mesma aplicação. Pelo fato de acessar os objetos IB_Connection e IB_Transaction de maneira embutida através das propriedades encontradas no TIBODataset, é possível utilizar a mesma conexão e transação nesses tipos “híbridos” de aplicativos.

Isso se torna particularmente útil quando você possui uma abordagem de migração de um aplicativo baseado na BDE para um baseado em controles nativos IBO. Isso é possível simplesmente utilizando uma ferramenta de conversão BDE-to-IBO nessa primeira fase.

Focando o Controle Global

Todos os componentes de acesso a dados do IBO são coordenados através de um outro componente de sessão, ao qual pertencem. Uma rede de monitoramento de eventos existe e constantemente mantém o registro de qual controle possui o foco, juntamente com qual declaração SQL, Dataset, Datasource, Transação e Conexão a ele estão ligados. Isso permite que controles como barras de botões fiquem “escutando” essa rede global de monitoramento e esperando por eventos que decidam qual o Dataset tem o foco no momento. O mesmo acontece para Transações e Conexões.

Isso é obtido configurando as propriedades AnnounceFocus e AllowFocus do TIB_DataSource e a propriedade ReceiveFocus de vários controles que estão habilitados a checar esse monitoramento global. Exemplos disso são as várias barras com controles, como por exemplo, o TIB_Navigator e a TIB_SearchBar.

Artigo Original: http://www.ibobjects.com **Jason Wharton** jwharton@ibobjects.com	
Tradução e adaptação: **Leonardo Mozachi Neto** leonardo@mozachi.com.br	**Comunidade Firebird de Língua Portuguesa** **Visite a Comunidade em:** **http://www.comunidade-firebird.org**

A Comunidade Firebird de Língua Portuguesa foi autorizada pelo Autor do Original para elaborar esta tradução.

[CFLP – Revisão de Texto por CSG e AA]