Capítulo 1: O BDE Administrador e os SQL Links

Interface Explorer

Quando o Delphi precisa acessar um banco de dados, ele o faz através dos serviços do Borland Database Engine (BDE). O BDE funciona da mesma maneira para ler dados armazenados localmente em arquivos DB e DBF, bem como para os mais sofisticados sistemas client/server.

Muitos anos atrás, a Borland percebeu um problema com o acesso a banco de dados em seus softwares. Eles produziram vários produtos que eram usados para acesso a bancos de dados, mas cada um tinha um método diferente para conectar-se e utilizar os dados. A Borland percebeu que uma abordagem de conectividade de banco de dados unificada forneceria muitas vantagens. Então decidiram criar um novo software que deveria abstrair toda a funcionalidade de um banco de dados dentro de um engine.

Há vários anos atrás, a Borland vendia um Paradox engine para programadores. Era composto pela parte de acesso do DBMS Paradox e foi desenvolvido para ser usado em programas que precisavam ler e gravar informações em tabelas Paradox (.DB). O BDE substituiu este engine, mas adicionou funcionalidade para conexão com outros tipos de bancos de dados. O BDE também está disponível separadamente do Delphi, caso você precise de uma performance sólida em um ambiente de desenvolvimento que não seja da Borland.

Ter todas as ações mais comuns de banco de dados dentro de um pedaço de software produz muitos efeitos positivos. Um único conjunto de drivers permite a melhoria de um driver em particular sem ter que reivindicar a roda cada vez que um novo pacote tem acesso a um banco de dados. Isto significada que o acesso a banco de dados pode ser atualizado sem ter que atualizar um pacote de software inteiro. Se você instalar uma nova versão do BDE em um sistema com uma versão antiga do Paradox pode imediatamente beneficiar-se dos novos drivers.

O conceito de driver unificado ambém salva você do armazenamento permanente que desaparece quando você tem muito código redundante. Um sistema com Paradox, Delphi e BDE requer muito menos espaço em disco porque o Paradox e o Delphi não necessitam ter seus próprios engines de acesso a banco de dados. Hoje em dia onde cada nova geração de programas requerem mais espaço em disco rígido, cada bit economizado ajuda.

Por último, e contudo mais importante, usar um método de acesso a banco de dados comum garante que o código para acessar um tipo particular de banco de dados seja escrito apenas uma vez.

Este capítulo se concentrará em lhe dar uma introdução aos recursos e capacidades do DBE. Se você necessita maiores informações sobre acesso a banco de dados no Delphi, consulte o próximo capítulo.

O Utilitário BDE Administrador

Quando o Delphi se instala, ele cria um ícone ou atalho para o utilitário Database Engine Administrador, ou BDEAdmin.EXE. O Delphi 3 coloca todos os executáveis e arquivos de configuração no diretório Program Files\Borland\Common Files\BDE. Nas versões anteriores do Delphi estes arquivos eram colocados em um diretório chamado IDAPI (Delphi 1.0), nomeado depois de utilitário de configuração antigo. IDAPI significa Independent Database Application Programmer Interface, e dá este nome para muitos dos arquivos e diretórios no sistema. O BDEAdmin atua como sua interface para o BDE. O utilitário permite que você mude muitos aspectos de como o BDE trabalha. O BDE armazena suas informações de configurações no Registry do Windows. Depois que você mudar suas definições no utilitário BDE Administrator, você deverá gravar suas definições selecionando **Object | Apply** ou dando um clique sobre o botão Apply.

Página Database

Na página Database do BDE Administrator estão os aliases para os bancos de dados disponíveis. Os bancos do dados são exibidos em uma árvore hierárquica parecida com o Windows Explorer. Para visualizar seus aliases de bancos de dados, simplesmente dê um clique sobre o símbolo de mais próximo ao banco de dados. Uma vez exibido, a definição de cada alias pode ser facilmente visualizada ou modificada selecionando o alias para exibi-lo no painel a direita do BDE Administrator.

Aliases de Bancos de Dados

Os aliases de banco da dados dão a você a habilidade para mudar a maneira que trabalha com seus dados em uma maneira muito poderosa. No mundo das aplicações de negócios, os bancos de dados são realocados freqüentemente quando as organizações geram novos arquivos e movem suas operações de bancos de dados para servidores de bancos de dados dedicados. Alguns ambientes de desenvolvimento necessitam que você recompile sua aplicação toda vez que um banco de dados mude de local, ou para construir uma camada adicional de abstração dentro do seu código. O BDE automaticamente dá a você esta habilidade através do uso dos aliases. Antes de vermos em detalhes como os aliases do BDE trabalham, veremos um pouco além a natureza de dados e tabelas de bancos de dados.

Banco de Dados vs. Tabelas

Quando Wayne Radcliff projeto a estrutura de arquivo do dBASE nos tempos dos PC's-Compatíveis, ele usou a extensão DBF (Database File) para descrever isso. Este é um termo errado porque banco de dados significa uma coisa específica que não se aplica a arquivos CBF.

Cada arquivo DBF possui uma *tabela*, uma coleção ordenada de dados. Uma tabela divide informações entre colunas conhecidas como *campos* e linhas (Fields and Rows), que relacionados formam um *registro*. Esse termo "Banco de Dados" geralmente refere-se a coleção de tabelas que se relacionam a outras no mesmo caminho. Exceto para aplicações simples, o banco de dados consiste em múltiplas tabelas relacionadas e cada tabela contém vários registros e campos.

Creating Pseudo Databases

Agora que você entendeu que um banco de dados é composto por um conjunto de tabelas relacionadas, nós precisamos observar como implementar esse conjunto. Servidores de banco de dados como Interbase e Oracle têm uma estrutura de banco de dados que pode ser composta por múltiplas tabelas. Desenvolvedores de banco de dados muitas vezes precisam utilizar esse tipo de estrutura, mas em ambientes sem servidores geralmente falta um mecanismo para linkar múltiplas tabelas. Você pode criar seu próprio formato de banco de dados para conter todas as suas tabelas, mas depois você perderá a compatibilidade com outros programas que utilizam banco de dados mais antigos.

O BDE possui uma solução avançada para esse problema. Colocado um grupo de tabelas relacionadas em algum diretório do DOS, você pode associar essas tabelas juntas usando o BDE. Basta dar um alias, ou nome ao banco de dados, para o grupo de arquivos em um diretório particular, o BDE criou um conjunto de tabelas logicamente criadas. O BDE chama um conjunto como este de Pseudo Database. Um Pseudo-Database não possui características avançadas como encontramos em um Database Server, como preservar integridade referencial. De qualquer modo ele permite a você utilizar o nome do banco de dados como ponteiro para o local de seus dados.

Pseudo-Databases podem ser criados na página Database no BDE Administrator:
Para criar um novo alias, comece selecionando **Object | New** no menu principal.
Você será questionado a selecionar um tipo de driver na caixa de diálogo New Database Alias.

O driver Standard fornece acesso às tabelas DB e DBF. Dê um clique sobre o Ok para retornar à pagina Database. Agora, você precisará entrar com um nome para o alias no painel esquerdo.

Você precisa também definir a localização das tabelas no campo PATH. Certifique-se de gravar a configuração, selecionando Apply quando você estiver satisfeito com as definições. Agora quando você precisar pegar dados em uma tabela, você pode usar o alias ao invés do nome diretório.

Até agora, parece que nós simplesmente mudamos um nome - o diretório, para outro - o alias. Isto nos salva de problemas quando chegar a hora de mudarmos a localização do banco de dados. Nós podemos colocá-lo em um servidor de arquivos de rede, movendo-o para um disco rígido maior, ou até mesmo fazer a mudança para um sistema de banco de dados cliente-servidor. Você não terá que mudar ou recompilar seus programas após as tabelas terem sido removidas. Você precisará criar um novo ponteiro para o alias no BDE.

A meta dos aliases do BDE é criar um ponteiro para um ponteiro. Qualquer rotina de acesso a banco de dados do Delphi fará referência ao alias de banco de dados, o qual faz referência às tabelas. Agora, mudando o alias, você tem uma maneira rápida de achar seus dados quando eles forem movidos de um lugar para outro.

Você deverá, sempre que criar uma nova tabela de banco de dados ou acessar uma antiga, tentar referenciar-se a ela pelo alias ao invés do path. O Database Desktop, por exemplo, permite que você selecione alias como letras de drivers na caixa de diálogo

Save as. Então quando você estiver pronto para gravar um novo banco de dados, você pode selecionar letras de drivers para forçá-lo a ir para um diretório em particular ou usar o alias, ao invés disto.

Note que conforme você muda o alias no menu drop-down, o diretório também muda. Você pode especificar um alias ao invés de um diretório em qualquer lugar do Delphi quando referenciar-se a uma tabela.

Página Configuration

A página Configuration do BDE Administrator exibe e permite que você configure os drivers instalados usados pelo BDE para gerenciar tabelas. Estes drivers são divididos em dois tipos: drivers Native e OBDC. As definições para cada driver aparecerão na página Definition localizada no painel a direita quando o driver estiver selecionado.

Você pode modificar as definições de um driver simplesmente selecionando o driver e alterando a definição desejada na página Definitions. Contudo, você só pode modificar definições que não tenham rótulos em negrito.

Se você precisar adicionar um novo driver ODBC, selecione ODBC na página Configurations e então selecione **Object | New** no menu.

O outro item exibido na página Configuration é o objeto System.

Dentro deste item, nós podemos configurar as definições de sistema INIT e Format. As definições INIT de sistema são usados pelo BDE para iniciar uma aplicação e são armazenados no Registry do Windows. As definições Formats de sistema são usadas para definir parâmetros de Data (*Date*), Hora (*Time*) e Número (*Number*).

Os parâmetros Date que podem ser definidos no BDE Administrator determinam como valores string são convertidos para valores de data. Por exemplo, você pode decidir quando as datas são armazenadas com ano de quatro dígitos ou dois dígitos, e quando ou não itens de dia e mês são armazenados com um zero extra para valores de um único dígito.

Os parâmetros Time que podem ser definidos determinam como valores string serão convertidos para valores hora. Por exemplo, você pode decidir se eles são armazenados em formato de doze ou vinte e quatro horas e quando são armazenados segundos ou milisegundos.

Os parâmetros Number que podem ser definidos determinam como valores string são convertidos para valores numéricos. Por exemplo, você pode decidir qual separador Decimal ou Milhar será utilizado, e com quantas casas decimais serão armazenadas.

Distribuindo o BDE

Quando cria a maioria dos programas, o Delphi irá gerar um executável *stand-alone*.

Contudo, a natureza do BDE requer que ele trabalhe independentemente de todos os outros pacotes. O núcleo do BDE é formado por um conjunto de DLL's, as quais devem ser distribuídas com qualquer programa que acesse as funções do BDE. A Borland permite que você redistribua o BDE gratuitamente sem custos de licenças.

SQL Links

No início do capítulo, vimos os Native Drivers na página Configuration do BDE Administrator. Estes Native Drivers são os SQL Links. Eles são simplesmente arquivos .dll que auxiliam o BDE na conexão com os bancos de dados associados.

Você poderá aprender um pouco mais sobre cada um destes SQL Links através do sistema de Help do BDE Administrator. Selecione **Help | Content** no item de menu e BDE configuration topics.

Agora selecione BDE configuration settings, depois Driver settings. Vejamos o tópico de help para o InterBase SQL Link. Cada opção descreve o tópico relacionado. Muito embora estes tópicos possam ser extremamente úteis no entendimento do link entre o BDE e o banco de dados, utilizar o SQL Monitor pode oferecer informações mais substanciais para aplicações cliente/servidor, como veremos no próximo capítulo.

Capítulo 2: Utilizando o SQL Monitor

Visualizando um Log

O SQL Monitor no Delphi é utilizado para rastrear e monitorar o tempo de chamada entre aplicações cliente e servidores de banco de dados SQL remotos. Ele permite ver chamadas de instruções chave a um servidor remoto através de SQL Links ou ODBC. Estas instruções chave podem incluir SELECT, UPDATE, DELETE, etc. Para abrir o SQL Monitor selecione **Database | SQL Monitor** no menu principal do Delphi.

Abriremos a aplicação Order Entry Project e a executaremos para ver o SQL Monitor em ação:

O SQL Monitor nos permite ver a performance de instruções SQL em uma aplicação. Ele exibirá as instruções SQL geradas pelo BDE e poderemos utilizar o monitor para ver se as bibliotecas do cliente do banco de dados estão funcionando corretamente. A caixa de diálogo SQL Monitor consiste em duas janelas. A janela superior exibe as

instruções SQL na ordem em que são geradas. Próximo a cada instrução há um número de referência e a estampa da data e hora também é gerada para cada instrução. Esta informação de tempo é útil na otimização da performance de suas instruções e transações. A janela inferior no SQL Monitor exibe a instrução SQL inteira selecionada na janela superior. As duas janelas podem ser redimensionadas utilizando-se a barra 'divisora' localizada entre elas.

Existe também uma barra de ferramentas nesta caixa de diálogo:

O primeiro botão na barra de ferramentas é o botão Save Log. Com este comando podemos salvar o Log SQL gerado como um arquivo texto (.TXT) para ser utilizado e referenciado posteriormente.

O segundo botão na barra de ferramentas é o botão Copy. Este botão copia a instrução SQL selecionada para o clipboard do Windows. O próximo botão é utilizado para limpar a janela do SQL Monitor. Próximo ao botão Clear está o botão Pause Trace. Este botão é útil se você quiser interromper a geração de instruções SQL para talvez ver uma instrução específica. O próximo botão é o botão Always on Top. Este botão aloja o SQL Monitor na frente de qualquer outra janela aberta. O último botão na barra de ferramentas é o botão Trace Options. Este botão abre a caixa de diálogo Trace Options.

O SQL Monitor lhe dá a flexibilidade de escolher qualquer combinação de opções que você queira monitorar. A seção seguinte discutirá as opções de Trace Options.

Configurando Opções de Trace Options

Para selecionar as opções de Trace que você queira monitorar, selecione o comando de menu **Options | Trace Options** ou dê um clique no botão Trace Options.

Existem nove categorias que você pode monitorar com o SQL Monitor:

- Prepared Query Statements - Instruções a serem enviadas ao servidor
- Executed Query Statements - Instruções a serem executadas pelo servidor
- Statement Operations - Qualquer operação efetuada (FETCH, EXECUTE, etc.)
- Connect/Disconnect - Qualquer operação associada com conexão/ desconexão com um banco de dados.
- Transactions - Operações de transação (BEGIN, COMMIT, ROLLBACK)
- Blob I/O - Operações em tipos de dados Blob (GET BLOB HANDLE, STORE BLOB, etc.)
- Miscellaneous - Operações diversas
- Vendor Errors - Mensagens de erro retornadas pelo servidor de banco de dados
- Vendor Calls - Chamadas de funções da API ao servidor

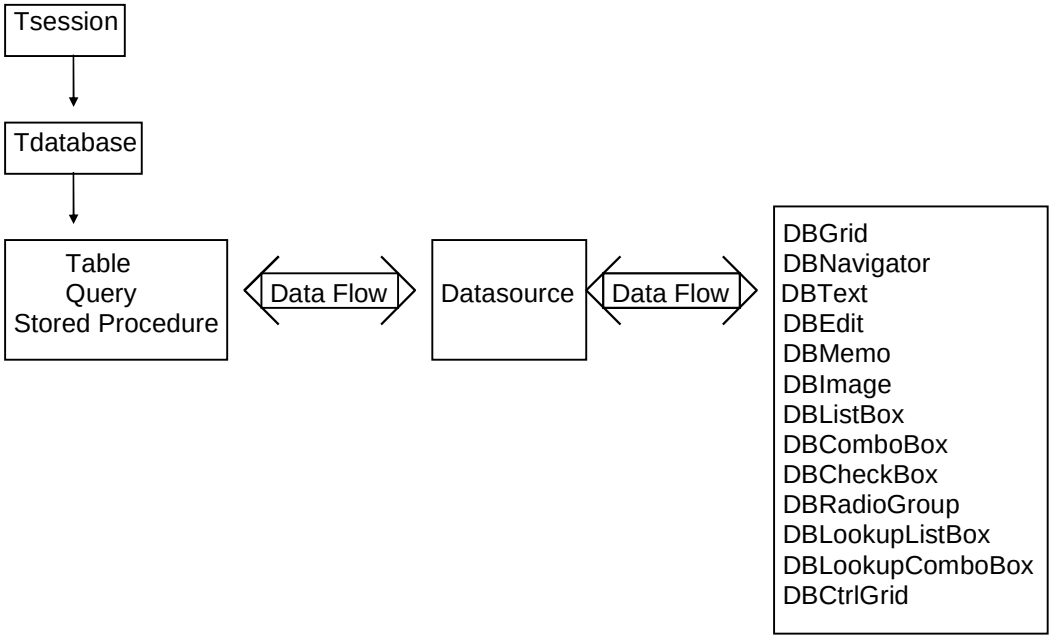
Utilizando diferentes combinações destas opções, o SQL Monitor pode se tornar uma ferramente importante. Por exemplo, vamos remover todas as opções na caixa de diálogo Trace Options exceto Executes Query Statements e Transactions.

Agora vamos executar o form Customer e visualizar as alterações no SQL Monitor.

Capítulo 3: Usando Sessions

Componentes TSession

O sistema para conexão de bancos de dados envolve cinco tipos de componentes: Session, Database, Dataset, DataSource e controles Dataaware. Estes compenentes relacionam-se uns com ou outros como exibido:



Antes de começarmos a falar em detalhes sobre o componente TSession, discutiremos alguns dos segredos do uso de tabelas Paradox em um ambiente rede. Existe um driver Paradox instalado com a configuração do BDE. Este driver contém uma opção chamada NetDir que especifica a localização do arquivo de controle de rede, PDOXUSR.NET. Este arquivo é necessário para permitir que várias aplicações e usuários compartilhem dados das mesmas tabelas Paradox em um drive de rede. Todas as aplicações que acessam estas tabelas devem especificar o mesmo diretório para este arquivo de controle de rede. A configuração para NetDir é usada por todas as aplicações Delphi que acessam tabelas DB. A verdade é que esta configuração que existe é bem limitada, especialmente na combinação de múltiplas aplicações, banco de dados, e seus ambientes. Isto é mais um fator para desenvolvedores que necessitam instalar aplicações em máquinas que já possuem esta configuração definida.

O componente TSession alivia o problema da configuração deste valor em várias máquinas. Ele contém várias propriedades para controlar o comportamento da sua

aplicação. A propriedade que irá ajudar-nos nesta situação é NetFileDir (discutida em detalhes mais tarde neste capítulo). Ele contém uma string que especifica o diretório para o arquivo PDOXUSRS.NET e toma precedência sobre a configuração BDE. Outra vantagem no uso deste componente é que ele encapsula as configurações na aplicação ao invés de forçá-las a basear-se em alguma coisa externa.

A principal função do componente TSession é gerenciar grupos de conexão de banco de dados com uma aplicação. O componente TSession isola um conjunto de operações de acesso a banco de dados, tais como conexões de banco de dados, sem a necessidade de iniciar outra instância da aplicação. Embora mais de um componente TSession possa ser usado em uma aplicação de banco de dados, normalmente um é suficiente. Na Figura 3.1, o componente TSession é um componente necessário para a conexão a um banco de dados. Contudo, você pode provavelmente escrever várias aplicações de banco de dados sem nunca usar um componente TSession. Na verdade, você estava usando o componente TSession e não sabia disso. Sempre que a unit DB.PAS for usada em uma aplicação, o Delphi automaticamente cria um componente padrão TSession acessível globalmente e chamado Session.

Para criar um exemplo que exibirá como o componente TSession está sendo criado automaticamente pelo Delphi, crie uma nova aplicação e coloque apenas um componente TDatabase no form. No manipulador de evento OnCreate do form, coloque o seguinte código:

```
Session.NetFileDir := 'C:\Borland Courseware\Delphi 3 CS Techniques\';
```

Agora coloque um breakpoint nesta linha e execute a aplicação. Lembre-se que nós não colocamos um componente TSession manualmente no form.

Quando a aplicação parar, execute a linha de código e então acesse a caixa de diálogo Watch List usando o comando de menu **View | Watches**.

Adicione o seguinte watch ao Watch List:

```
Session.NetFileDir
```

Uma vez que o Delphi criou um componente TSession chamado Session, qualquer propriedade ou evento associado com este componente TSession padrão podem ser implementados em tempo de execução. Em nosso exemplo, definimos o NetFile Dir do componente TSession padrão como 'C:\Borland Courseware\Delphi 3 CS Techniques\'. Na próxima sessão, discutiremos o uso das propriedades NetFileDir e PrivateDir.

Propriedades do TSession

Nesta sessão, discutiremos certas propriedades que fazem do componente TSession um link vital na conexão de banco de dados. As propriedades que veremos são as seguintes: PrivateDir, NetFileDir, KeepConnections, SQLHourGlass, ConfigMode e TraceFlags.

PrivateDir

A propriedade PrivateDir define o diretório no qual serão armazenadas os arquivos de tabelas temporárias geradas pelo BDE. Ordinariamente, este valor só é definido em tempo de execução para que o disco rígido local do usuário possa ser usado para armazenar os arquivos temporários. Contudo, ele é uma propriedade Published e pode ser definida em tempo de desenvolvimento.

Uma razão importante para usar a propriedade PrivateDir é para aumentar a performance da aplicação. Definindo PrivateDir para armazenar os arquivos de tabelas temporárias localmente pode reduzir o tráfego de rede, com isso aumentando substancialmente a performance da aplicação. Ela também evita que tabelas temporárias de vários usuários interfiram em tabelas de outros. Se nenhum valor for especificado para PrivateDir, o BDE automaticamente armazena os arquivos temporários no diretório atual do momento em que o BDE for iniciado.

NetFileDir

NetFileDir define o diretório no qual o arquivo de controle de rede do BDE, PDOXUSRS.NET, estará contido. Como PrivateDir, você pode definir NetFileDir em tempo de desenvolvimento.

Você também pode definir ou mudar NetFileDir no código em tempo de execução.s código a seguir define o NetFileDir do session padrão para o diretório no qual sua aplicação está sendo executada.

```
Session.NetFileDir := ExtractFilePath(ParamStr(0));
```

Uma nota importante é que NetFileDir só pode ser mudada quando todos os arquivos Paradox na aplicação estiverem fechados. Também, se NetFileDir for mudada em tempo de execução, tenha certeza que ela aponte para um diretório que seja compartilhado pelos usuários da rede. Outra nota é que as propriedades NetFileDir e PrivateDir não podem apontar para o mesmo diretório.

KeepConnections

A propriedade KeepConnections é uma propriedade do tipo Boolean que realiza uma tarefa interessante. KeepConnections especificam quando ou não um componente TDatabase temporário, criado no contexto de uma sessão, mantém uma conexão de servidor de banco de dados mesmo se não houverem datasets abertos associados com o componente database.

Isto simplesmente significa que se um componente TDatabase for criado em tempo de execução, a propriedade KeepConnections deverá especificar quando o banco de dados permanece ativo, se não houverem datasets conectados ao TDatabase. A configuração padrão para a propriedade KeepConnections do TSession é True. Se KeepConnections for definido como False e todos os datasets associados com um determinado database estiverem fechados, o componente TDatabase irá derrubar a conexão e liberar os recursos de sistema alocados para ela. Se isto ocorrer, o componente TDatabase deve ser re-conectado antes que qualquer dataset associado com o TDatabase possa ser aberto.

O componente TDatabase também tem uma propriedade KeepConnections que será discutida no capítulo sobre TDatabase.

SQLHourGlass

A propriedade SQLHourGlass indica simplesmente quando ou não o cursor do mouse deve mudar para uma ampulheta durante operações do BDE. Se SQLHourGlass estiver definido como True, o cursor de ampulheta aparecerá quando o BDE estiver em progresso. Se SQLHourGlass estiver definido como False, o cursor permanecerá o mesmo durante operações do BDE.

ConfigMode

A propriedade CofigMode especifica como o Borland Database Engine (BDE) deve manipular aliases para a sessão. Você pode ter notado que ConfigMode não é uma propriedade disponível no Object Inspector e só pode ser definida em tempo de execução. Isto é graças ao fato de que ConfigMode só funciona com os aliases criados usando os métodos AddAlias ou AddStandardAlias dentro do contexto de uma sessão.

Existem três ConfigModes disponíveis. Os três modos são os seguintes:

Modo	Definição
cfm Virtual (Padrão)	Todos os aliases no arquivo de configuração do BDE, os armazenados permanentemente no BDE e aliases locais a sessão estão disponíveis a sessão.
cfmPersistent	Apenas os aliases do arquivo de configuração do BDE, ou aqueles que são adicionados ao armazenamento persistente estão disponíveis para a sessão.
CfmSession	Apenas os aliases criados dentro desta sessão estão disponíveis para a sessão.

TraceFlags

A última propriedade do componente TSession que discutiremos é TraceFlags. A propriedade TraceFlags é usada para especificar quais operações do banco de dados o SQL Monitor deverá analisar para o componente TSession. Se dermos uma olhada na caixa de diálogo Trace Options no SQL Monitor, veremos todas as opções disponíveis para TraceFlags.

Uma vez que TraceFlags não é uma propriedade disponível no Object Inspector, a única maneira de definir estas opções de trace é em tempo de execução. O Delphi tornou isto simples criando um conjunto (set) chamado TraceFlags. O código a seguir demonstra como definir Trace Options para Executed Query Statements e Transactions.

```
Session1.TraceFlags := [tfQExecute, tfTransact];
```

A tabela a seguir lista os valores associados com a propriedade Trace Flags:

Valores	Definição
tfQPrepare	Monitora instruções de Prepare.
tfQ Execute	Monitora instruções de ExecSQL.
tfError	Monitora mensagens de erro do servidor. Tais mensagens podem incluir um código de erro.
tfStmt	Monitora todas as instruções SQL.
tfConnect	Monitora operações de conexão de desconexão de banco de dados, incluindo alocação e liberação de handles de conexão.
tfTransact	Monitora instruções de transações, tais como StartTransaction, Commit e Rollback.
tfBlob	Monitora operações em tipos de dados blob.
tfMisc	Monitora qualquer instrução não coberta pelas outras opções de flag.
tfVendor	Monitora chamadas diretas a funções de API do servidor de banco de dados.
tfDataIn	Monitora dados recebidos de um servidor.
tfDataOut	Monitora dados enviados para um servidor.

Uma coisa importante a que se notar: se você definir Trace Options em tempo de execução, as opções não aparecerão marcadas no SQL Monitor, mesmo depois que a linha de código onde a propriedade TraceFlags for definida tiver sido executada. Contudo, os resultados no SQL Monitor irão demonstrar claramente que aquelas opções estão sendo realizadas.

Os Manipuladores de Evento de TSession

O TSession só tem dois eventos associados a ele. Eles são OnPassword e OnStartUp. Não discutiremos o manipulador de evento OnPassword uma vez que só lida com tabelas Paradox. Contudo, veremos detalhadamente o m
manipulador de evento OnStartUp.

OnStartUp

Quando um componente TSession é ativado, várias ações interessantes ocorrem, muitas das quais não possuímos controle. Uma ação que podemos usar é o evento OnStartUp. O evento OnStartUp é disparado quando a propriedade Active do seu componente TSession correspondente for definida como True. Isto pode ser muito útil em aplicações Client/Server. Para o momento, podemos incluir alguma funcionalidade

que ocorra antes do evento OnLogin do TDatabase na nossa aplicação. Se ambos os componentes TSession e TDatabase não estiverem ativos e a propriedade Connected do componente TDatabase tiver sido de repente definida como True, o manipulador de evento OnStartUp do TSession será disparado, inicialmente.

Métodos TSession

O ultimo tópico a discutir neste capítulo são os Métodos do TSession.

Embora não vejamos cada método disponível no TSession, falaremos rapidamente sobre os métodos mais usados do TSession. Alguns dos métodos que veremos são: AddAlias, DeleteAlias, IsAlias, ModifyAlias e SaveCofigFile.

AddAlias

O método AddAlias adiciona um alias BDE em tempo de execução para um servidor de banco de dados SQL que está associado com a sessão atual. O método AddAlias requer três parâmetros. Estes três parâmetros são Nome, Driver e um String List de parâmetros para o alias. É importante notar que este alias só estará disponível para o TSession específico. Qualquer dos componentes não conectados ao TSession que cria o novo alias não será capaz de usar o alias. Vamos ver um pedaço de código que demonstra como usar o método AddAlias. Colocaremos este código no evento OnCreate de um form em uma nova aplicação.

```
Procedure TfrmAddAlias.FormCreate(Sender: Tobject);
var
     AliasInfo: TStringList;
begin
     AliasInfo := TStringList.Create;
     try
          with AliasInfo do
          begin
               Add('SERVER NAME=C:\Delphi Training Class\Sample
                   Data\ORDERENTRY.GDB');
               Add('USER NAME=SYSDBA');
          end;// with..do
          Session1.AddAlias('DelphiServerData', 'INTRBASE',
               AliasInfo);
     finally
          AliasInfo.Free
     end;// try..finally
end;
```

Neste exemplo, novo alias recebe o nome de DelphiServerData. O Driver usado foi INTRBASE. Os parâmetros do alias foram armazenados em um TStringList chamado AliasInfo. SERVER NAME e USER NAME para o banco de dados foram colocados nesta variável AliasInfo.

Você vê algum problema que possa estar associado com este exemplo? Um problema é que o alias nunca ‘é destruído.

Se não destruirmos este alias, toda vez que este form for criado, uma exceção ocorrerá.

Existem várias maneiras de resolver este problema. Uma maneira é deletar o alias no BDE Administrator ou no SQL Explorer. Outra maneira seria usar o método DeleteAlias.

Uma importante observação é que o método AddAlias não grava o novo alias no arquivo de configuração do BDE. Se você quiser que o novo alias seja gravado no arquivo de configuração do BDE, o método SaveConfigFile tem que ser chamado. O mesmo será discutido mais tarde neste capítulo.

DeleteAlias

Como aprendemos na sessão anterior, podemos criar um alias em tempo de execução. Também é possível apagar um alias em tempo de execução. O método usado para fazer isto é DeleteAlias. O método DeleteAlias aceita uma string que designa qual alias deve ser apagado. Este método não apaga o alias do arquivo de configuração do BDE. Se você quiser apagar completamente um alias, primeiro chame o método DeleteAlias seguido pelo SaveConfigFile.

Para o próximo exemplo, estaremos usando o form com o código criado na sessão sobre AddAlias. Depois de adicionar um botão ao form, colocaremos o método DeleteAlias dentro do manipulador de evento OnClick. Quando o usuário der um clique sobre o botão, o alias que criamos será apagado.

```
Procedure TfrmAddAlias.btnDeleteAliasClick(Sender: Tobject);
begin
     Session1.DeleteAlias(‘DelphiServerData‘);
end;
```

Depois da aplicação ter sido executada e o método DeleteAlias tiver sido executado, veremos que o alias DelphiServerData foi apagado.

IsAlias

Existem tantas instâncias quando você está criando e apagando um alias, que você precisa saber se um alias em específico é válido. O método IsAlias permite resolver este problema. O método IsAlias aceita uma string e retorna um valor Boolean baseado nesta string. O valor da string que é passado ao método ISAlias é o alias que você está tentando verificar.

Se o método IsAlias retorna um valor True, o alias é válido. Se o método IsAlias retorna False, o alias então foi apagado ou nunca mais foi criado.

Para demonstrar o método IsAlias, usaremos o manipulador de evento btnDeleteAliasClick que criamos anteriormente. No manipulador de evento, iremos adicionar uma instrução if..then que verifica se o alias ‘DelphiServerData’ é válido. Se ele for um alias válido, iremos então apagá-lo. O código deverá parecer com o seguinte:

```
procedure TfrmAddAlias.btnDeleteAliasClick(Sender: Tobject);
```

```
begin
     if (Session1.IsAlias('DelphiServerData')) then
     begin
          Session1.DeleteAlias('DelphiServerData');
     end;
end;
```

ModifyAlias

Existem situações nas quais os parâmetros de alias precisam ser mudados em tempo de execução. Ao invés de apagar o alias e criar um novo alias com parâmetros diferentes, o Delphi nos dá uma alternativa mais fácil. Podemos usar o método ModifyAlias para adicionar ou mudar os parâmetros para um alias específico do BDE. Para fazer isto, chamamos o método ModifyAlias e passamos a ele o nome do alias e lista de mudanças. O nome do alias é simplesmente uma string - a lista de mudanças é do tipo TString.

Para outro exemplo, criaremos um novo botão no form que modificará o alias DelphiServer Data. A modificação será feita no parâmetro USER NAME.

```
Procedure TfrmAddAlias.btnModifyAliasClick(Sender: Tobject);
var
     AliasInfo: TStringList;
begin
     AliasInfo := TStringList.Create;
     try
          AliasInfo.Add('USER NAME=CWILLIAMS');
          Session1.ModifyAlias('DelphiServerData', AliasInfo);
     finally
          AliasInfo.Free;
     end;// try..finially
end;
```

SaveConfigFile

SaveConfigFile é um método poderoso, mas perigoso. O seu propósito é transferir quaisquer mudanças na configuração do BDE, atualmente na memória, para o arquivo de configuração do BDE. Como temos visto com outros métodos (AddAlias, DeleteAlias, etc.), mudanças podem ser feitas nos aliases, contudo, estas mudanças não são feitas permanentemente. Quaisquer mudanças feitas com o método SaveConfigFile serão permanentemente gravadas no arquivo de configuração do BDE. Você deve ser bem cuidadoso quando usar este método. Se você usar o método DeleteAlias e chamar o SaveConfigFile logo em seguida, o alias será apagado permanentemente. O código a seguir demonstra a sintaxe necessária no uso do método SaveConfigFile.

Comentario: S feitas

```
Session1.SaveConfigFile;
```