Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação

C++ Builder

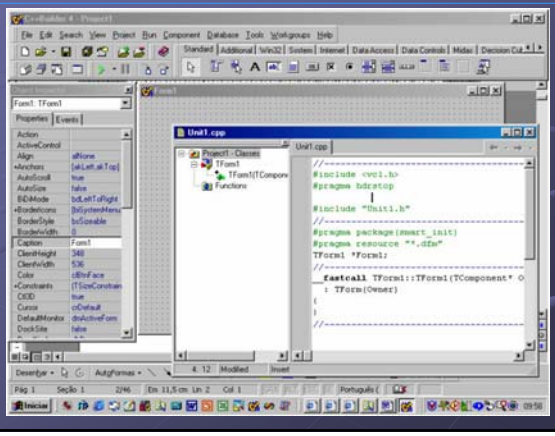
SCE000 – Laboratório de Bases de Dados

Joselene Marques

Roteiro

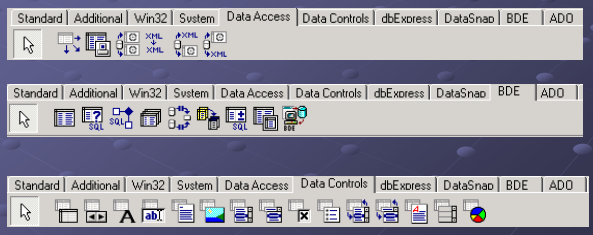
1. **Ambiente Desenvolvimento Integrado (IDE)**
2. **Aplicações de bancos de dados**
   - Usando componentes TDataSets, TTable,TDataSource, TQuery, TDBGrid, TDBEdit, TDBText
   - BDE Administrator
   - Database Desktop para criar novas tabelas
   - Database Explorer para visualizar o banco de dados
3. **Criando um formulário de Cadastro**
   - Consultas SQL dinâmicas (com parâmetros)
   - Comandos SELECT, INSERT, UPDATE e DELETE
4. **Componente Database**
5. **SGBD Remoto**
6. **Criando um formulário de Login**

Ambiente Desenvolvimento Integrado (IDE)



Aplicações de bancos de dados

- Borland Database Engine (BDE)
- Classes de componentes: Data Access, BDE e Data Controls

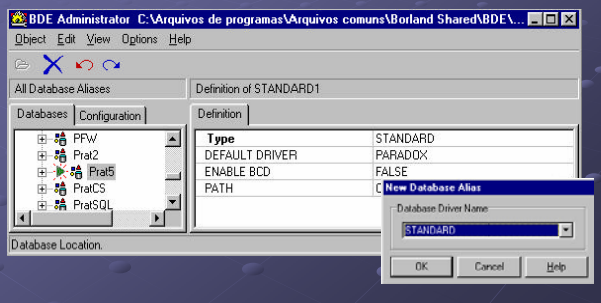


Configurando o BDE

- BDE Administrator (BDEAdmin.EXE)
  - configurar os aliases utilizados e alterar o ambiente do Engine (IDAPI32.CFG)
- Database Desktop (não aconselhável em versões recentes)
  - visualizar criar e modificar tabelas Paradox e dBase
  - executar operações restritas em outros bancos de dados
- Database Explorer / SQL Explorer (Enterprise)
  - visualizar e manter bancos de dados (remotos ou locais, com restrições)
  - trabalhar com aliases do BDE e com metadados

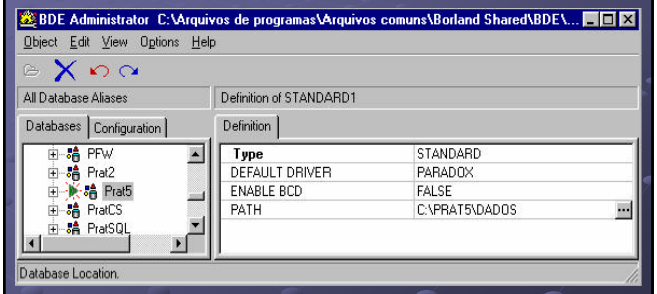
BDE Administrator

1) Iniciar | Programas | Borland C++Builder | BDE Administrator (ou Painel de Controle do Windows).
2) Escolha no menu Object a opção New.



BDE Administrator

3) Configure o nome do alias e o PATH
4) Escolha o menu Object | Apply



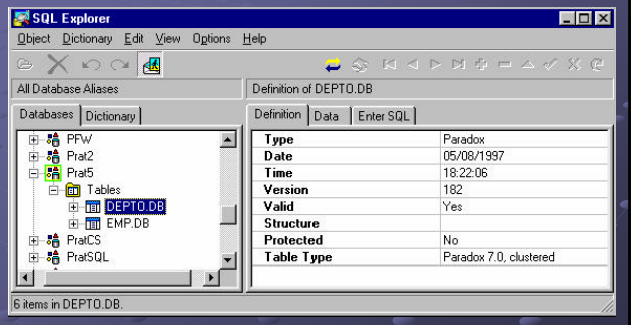
DataBase Desktop

- Menu Iniciar | Programas | Borland C++Builder | DataBase Desktop

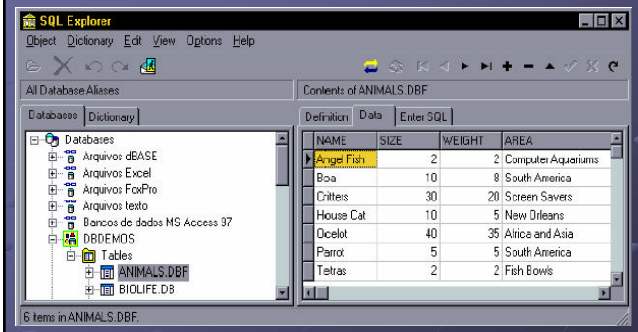
- Fornece uma interface simples e completa para configuração, definição e manipulação de tabelas de bancos de dados Paradox e dBase.

Database Explorer

No C++Builder, escolha o menu Database | Explore

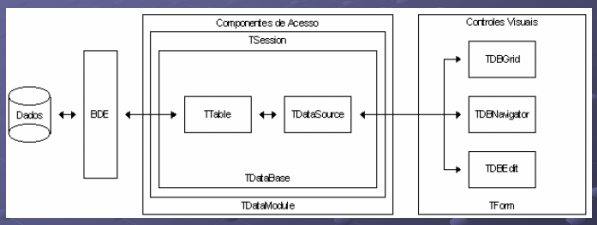


Database Explorer



Aplicações de bancos de dados

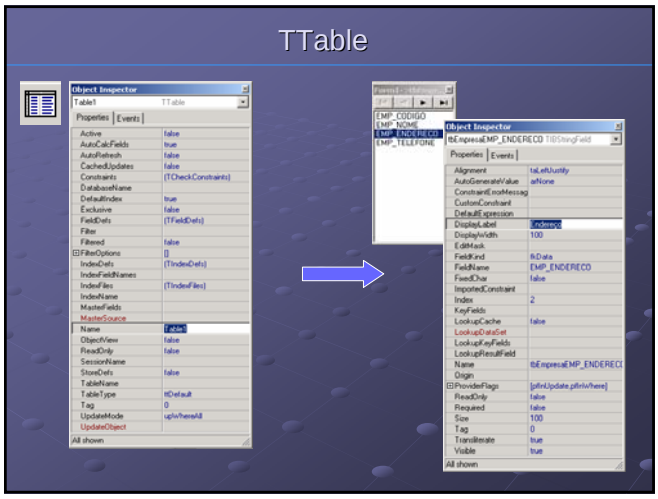
BDE + Aplicação



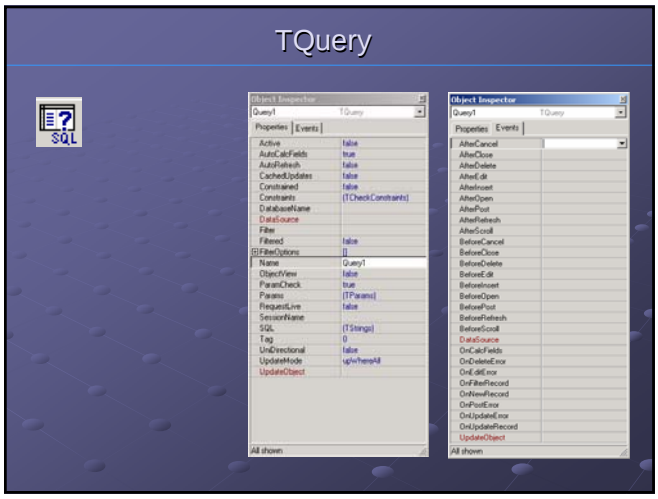
TDataSets

- Table da classe TTable
- Query da classe TQuery
- StoredProc da classe TStoredProc.

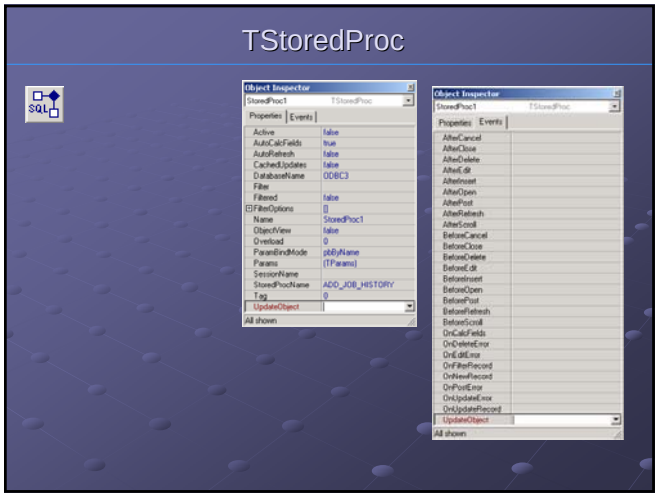
TTable



TQuery



TStoredProc



TDataSets

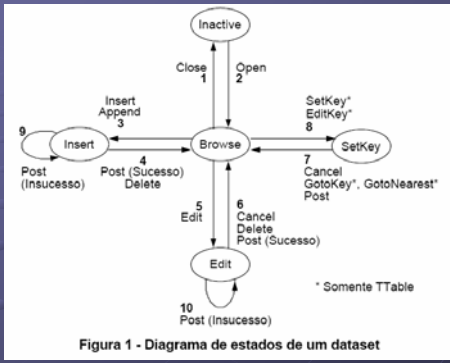


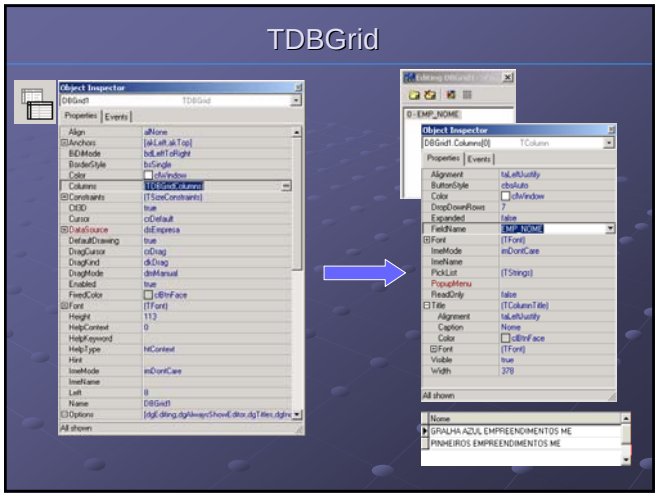
Figura 1 - Diagrama de estados de um dataset

Conjunto de estados do dataset

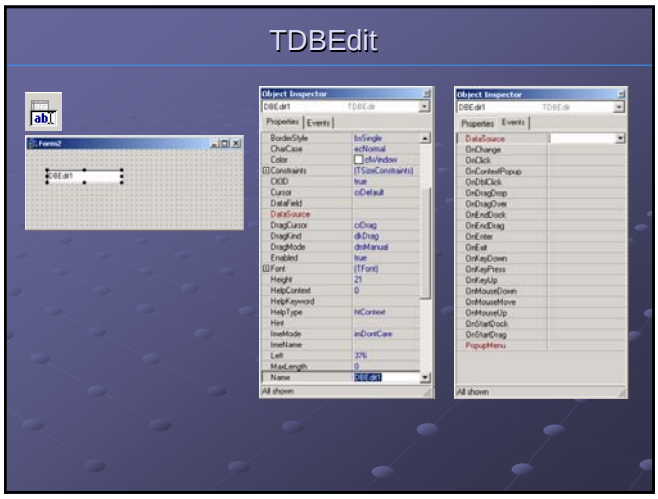
Inactive	Quando o dataset encontra-se fechado.
Browse	O estado no qual o dataset se encontra quando o mesmo é aberto e permanece a maior parte do tempo de seu uso. Registros podem ser lidos, mas não alterados ou inseridos.
Edit	Permite que a linha (registro) corrente seja editada (alterada).
Insert	Permite que uma nova linha seja inserida e alterada. Após uma chamada ao método Post uma nova linha é gravada na tabela e o dataset volta ao estado de Browse.
SetKey	Permite que sejam atribuídos valores aos campos que compõem o índice atualmente em uso pelo dataset, possibilitando que os métodos GoToKey e GoToNearest realizem buscas na tabela. Este estado e métodos só estão disponíveis em componentes da classe TTable.
CalcFields	Estado que ocorre quando um campo calculado está sendo atualizado.

DataSource

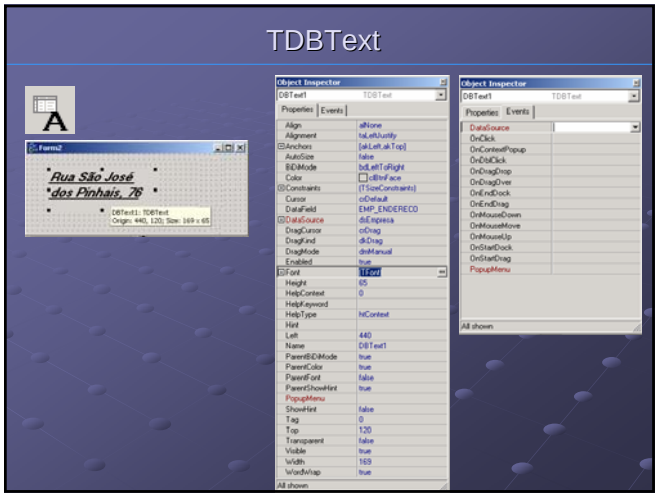
TDBGrid



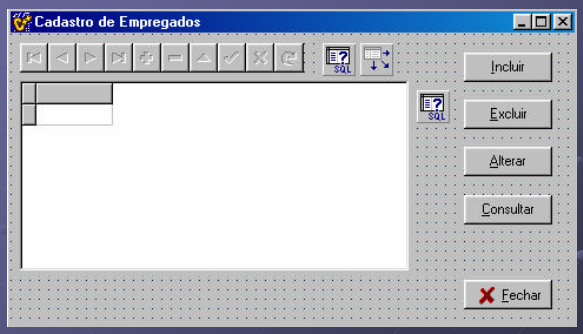
TDBEdit



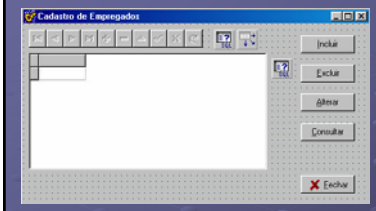
TDBText



Criando um Formulário de Cadastro



Criando um Formulário de Cadastro

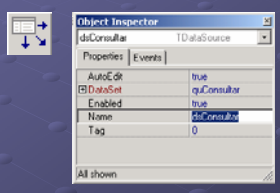


Componentes
- ✓ 4 botões de comando (Button)
- ✓ 1 botão BitBtn
- ✓ 2 componentes Query
- ✓ 1 componente DataSource
- ✓ 1 DBGrid
- ✓ 1 DBNavigator.

Alterações
- ✓ Alterar o nome do DBGrid para “GridConsulta”.
- ✓ Alterar os nomes (propriedade Name) dos botões de comando para “btIncluir”. “btExcluir”, “btAlterar” e “btConsultar”.
- ✓ Configure a propriedade Kind do BitBtn para bkCancel e Name para “btFechar”.

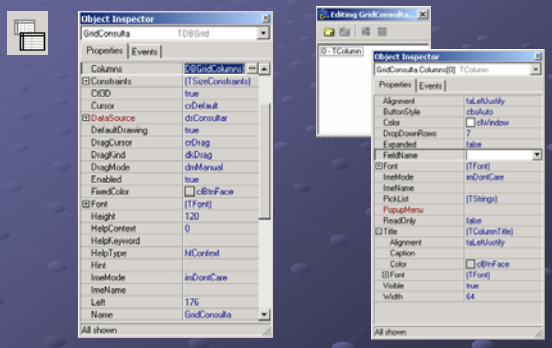
Criando um Formulário de Cadastro

1) Dê o nome de “quConsultar” ao componente Query localizado acima do componente DBGrid
2) Dê o nome de “quExcluir” ao Query localizado ao lado do botão “Excluir”.
3) Altere o nome do DataSource para “dsConsultar” e ligue ao Query quConsultar.

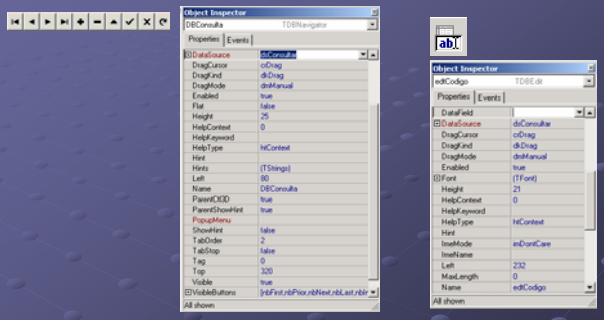


Criando um Formulário de Cadastro

4) Efetue a ligação dos componentes com o DataSource



Criando um Formulário de Cadastro



5) Altere a Caption do formulário para “Cadastro de Empregados” e seu nome para J_CadEmp. Salve o projeto dando o nome de CadEmp.cpp para o formulário e PratSQL.bpr para o projeto

Criando um Formulário de Cadastro

6) Configure o componente quConsultar: propriedade DatabaseName para o valor definido no BDE;

7) Configure a propriedade SQL, Exemplo:

SELECT CodEmp, Nome FROM EMPREG

8) Ative o editor de campos (Fields Editor) do componente quConsultar e inclua os campos que irão aparecer na lista (retornados pelo comando SELECT).

9) Altere a propriedade DisplayLabel dos campos ou configure no DBGrid

10) Programe o evento OnActivate do formulário com o comando quConsultar->Open();

11) Programe o evento OnClick do botão Fechar: Close();

Criando um Formulário de Cadastro

Excluindo

12) Configure o componente quExcluir: propriedade DatabaseName para o valor definido no BDE;

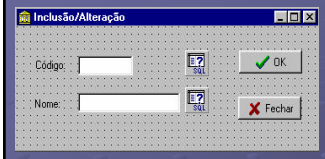
13) Configure a propriedade SQL, Exemplo:

```
DELETE FROM EMPREG WHERE CodEmp = :Codigo
```

20) Programe o evento OnClick do botão "Excluir" com as seguintes linhas de comando:

```
if (Application->MessageBox( "Excluir este empregado ?",
"Confirma Exclusão", MB_YESNO | MB_ICONQUESTION) == IDYES) {
      quExcluir->ParamByName("Codigo")->AsInteger =
            GridConsulta->Fields[0]->AsInteger;
      quExcluir->ExecSQL();
      quConsultar->Close();
      quConsultar->Open();
}
```

Criando um Formulário de Cadastro



Componentes

- ✓ 2 componentes Edit: edCodigo e edNome
- ✓ 2 componentes BitBtn: btOK (propriedade Kind = bkOK) e btFechar (propriedade Kind = bkCancel)
- ✓ 2 componentes Query (quIncluir e quAlterar)

21) Configure o componente quIncluir: propriedade DatabaseName

22) Configure a propriedade SQL, Exemplo:

```
INSERT INTO Empreg (CodEmp, Nome) VALUES (:Codigo,:Nome)
```

23) Configure o componente quAlterar: propriedade DatabaseName

24) Configure a propriedade SQL, Exemplo:

```
UPDATE Empreg SET Nome = :Nome WHERE CodEmp = :Codigo
```

Criando um Formulário de Cadastro

25) Altere a Caption do formulário para "Inclusão/Alteração" e chame-o "J_IncluiAltera". Salve-o com o nome IncAlt.cpp.

26) Programe o evento OnClick do botão "OK" com as seguintes linhas de comando:

```
if (Caption == "Inclusão") {
    quIncluir->Params->Items[0]->AsInteger=StrToInt(edCodigo->Text);
    quIncluir->Params->Items[1]->AsString = edNome->Text;
    quIncluir->ExecSQL();
} else {
    quAlterar->Params->Items[0]->AsString = edNome->Text;
    quAlterar->Params->Items[1]->AsInteger = StrToInt(edCodigo->Text);
    quAlterar->ExecSQL();
}
```

- Use também: `ParamByName("Campo")`

Criando um Formulário de Cadastro

Chamando a inclusão/alteração

27) Volte para o formulário principal.

28) Programe o evento OnClick do botão “Incluir” do formulário principal com as seguintes linhas de comando:

```
J_IncluiAltera->edCodigo->Clear();
J_IncluiAltera->edNome->Clear();
J_IncluiAltera->Caption = "Inclusão";
J_IncluiAltera->edCodigo->ReadOnly = false;
J_IncluiAltera->ShowModal();
quConsultar->Close();
quConsultar->Open();
```

Criando um Formulário de Cadastro

29) Programe o evento OnClick do botão “Alterar” do formulário principal com as seguintes linhas de comando:

```
J_IncluiAltera->edCodigo->Text = GridConsulta->Fields[0]->AsString;
J_IncluiAltera->edCodigo->ReadOnly = true;
J_IncluiAltera->edNome->Text = GridConsulta->Fields[1]->AsString;
J_IncluiAltera->Caption = "Alteração";
J_IncluiAltera->ShowModal();
quConsultar->Refresh();
```

Criando um Formulário de Cadastro

- Para que o formulário de inclusão e alteração possa ser chamado a partir do formulário principal, é necessário usá-lo no formulário principal, através do menu File | Include Unit Hdr. Escolha a Unit IncAlt.

30) Execute e teste a sua aplicação.

- Teste a aplicação, tentando incluir, alterar ou excluir algum registro através do DBGrid. Não é possível, pois o comando SELECT do SQL retorna valores somente de leitura.
- Configure a propriedade **RequestLive** do componente **quConsultar** para **True** (Nem sempre permitido – visões atualizáveis)

Criando um Formulário de Cadastro

Consultas Preparadas

- Previamente analisada, reescrita e planejada
- Criadas pelo método PREPARE
- Podem receber parâmetros que são substituídos na consulta quando é executada
- São armazenadas localmente (no processo servidor corrente), e somente vão existir enquanto durar a sessão do banco de dados.
- Cada cliente pode criar e usar a sua própria consulta preparada.

Melhoram desempenho:

- Quando um único processo servidor é usado para executar um grande número de consultas similares
- Para consultas com um planejamento ou uma reescrita complexos (junção de muitas tabelas, aplicação de várias regras)

Componente Database

- Proporciona controle adicional sobre fatores que são extremamente importantes em aplicações cliente/servidor:
  - conexão com o SGBD
  - processamento de transações
  - níveis de isolamento, etc.
- Não é obrigatório criá-lo. Se necessário, o C++ Builder cria um componente Database temporário (virtual)
- A propriedade AliasName é o nome de um alias do BDE existente.
- A propriedade DatabaseName é um alias interno ao programa.

SGBD remoto

- Possuir a versão Enterprise - drivers nativos (Oracle, SQL-Server, Sybase, Informix, DB2 e Interbase)
- Pode-se utilizar outros SGBDs, através de ODBC.

1) Criar o alias no BDE com o driver correspondente fazendo as configurações necessárias.

2) Instalar produtos específicos. Ex: no Oracle, deve-se instalar o SQL-Net

Criando um Formulário de Login

1) Crie um **Data Module** através do menu File | New Data Module.

2) Em Database (Data Access) configure a propriedade **AliasName** para “PratCS” e a propriedade **DatabaseName** para “BDCliente”.

➢BDCliente será utilizado nos componentes **DataSet** em substituição ao alias. Quando se usa o componente Database somente ele precisa se referenciar diretamente ao alias externo.

Criando um Formulário de Login

Obs. 1: Será solicitada a conexão com o banco de dados.

Obs. 2: Em caso de sucesso na conexão a propriedade Connected é configurada automaticamente para True.

➢ Para inibir a conexão com o banco, configure a propriedade **LoginPrompt** do Database para “False” e a propriedade **Params** com:

```
USERNAME=SYSDBA
PASSWORD=masterkey
```

➢ Não é um método seguro !

Criando um Formulário de Login

Customizando a janela de conexão

3) Crie um novo formulário de acordo com a figura abaixo:

4) Chame os componentes de edUsuário e edSenha.

5) Configure a propriedade PasswordChar do edSenha com * (asterisco) para ocultar a senha enquanto o usuário está digitando.

6) Altere a propriedade **Name** para J_Login, propriedade **Caption** para Conexão com o SGBD, propriedade **Position** para poScreenPosition e salve-o com o nome Login.cpp.

Criando um Formulário de Login

Opção 1 – alterando o evento OnLogin do Database

7) Configure o componente Database - LoginPrompt = "True"

8) Programe o evento OnLogin do Database:

```
J_Login = new TJ_Login(Application);
try {
    if (J_Login->ShowModal() == mrOk) {
        LoginParams->Values["USER NAME"] = J_Login->edUsuario->Text;
        LoginParams->Values["PASSWORD"] = J_Login->edSenha->Text;
    } else {
        Application->Terminate();
    }
} catch (...) {}
delete J_Login;
```

9) Faça com o Data Module usar o formulário de login - File | Include Unit Hdr.

Criando um Formulário de Login

10) Para tratar a exceção adequadamente, configure a propriedade Connected do Database = False e altere a programação do evento OnCreate do DataModule:

```
try {
    Database1->Open();
} catch (Exception &e) {
    Application->MessageBox("Usuário ou Senha são
    inválidos!", "Falha na conexão", MB_OK | MB_ICONHAND);
    Application->Terminate();
}
```

Criando um Formulário de Login

Opção 2 – Melhorando o Diálogo de Login

Permite ao usuário 3 tentativas

1) Para inibir o evento OnLogin do componente Database (opção 1), Através do Object Inspector, apague a chamada ao evento.

- Não é necessário apagar o código, apenas fazer com que ele não seja ativado.

2) Configure a propriedade LoginPrompt do Database para "False"

3) Substitua o evento OnCreate do Data Module por:

```
int tentativas = 1;
bool usrInvalido = true;
J_Login = new TJ_Login (Application);
while (usrInvalido) {
    if (J_Login->ShowModal() == mrOk) {
        Database1->Params->Values["USER NAME"] = J_Login->edUsuario->Text;
        Database1->Params->Values["PASSWORD"] = J_Login->edSenha->Text;
        try {
              Database1->Open();
              usrInvalido = false;
        } catch (...) {
              Application->MessageBox(
                    "Usuário ou Senha são inválidos! ",
                    "Falha na conexão", MB_OK | MB_ICONHAND);
              if (++tentativas > 3) break;
        }
    }
        else break;
    }
    delete J_Login;
    if (usrInvalido) {
        Application->ShowMainForm = false;
        Application->Terminate();
}
```

C++ Builder

Bibliografia:

"Práticas de C++ Builder", Leônidas Francisco de Lima Junior, Janeiro/2001

FIM