

Eventos e o Firebird

Anderson Haertel Rodrigues

Introdução:

Eventos é um recurso que o Banco de Dados Firebird disponibiliza aos usuários. Eventos como a própria palavra cita, são notificações que o Banco de Dados Firebird manda para a sua aplicação. A aplicação por sua vez resgatou o Evento enviado pelo Firebird e vai disparar e ou executar alguma rotina na aplicação. Podendo inclusive interagir com o Firebird através de uma execução de Stored Procedure! Interessante Não?!

Stored Procedures e Triggers:

Eventos só podem ser disparados por Stored Procedure (vulgo SP) e ou Trigger (vulgo Gatilho). Vou explicar de uma forma mais genérica o que são SP's e Trigger's.

Stored Procedure:

São “programas” em linguagem SQL que são compilados e armazenados no Banco de Dados Firebird. Um dos principais objetivos da SP é a performance, pois, a mesma é executada no servidor e retorna apenas os parâmetros e ou linha do ResultSet para a aplicação. Existem dois tipos de SP:

SELETÁVEIS;
EXECUTÁVEIS;

Stored Procedure Seletável “SELECT * FROM NOME_SP”. Tem a função de retornar linhas de Dados para a aplicação.

Stored Procedure Executável “EXECUTE PROCEDURE NOME_SP”. Tem a função de retornar parâmetros de alguma operação feita no Servidor.

Triggers:

São “programas” associados a reações de atualização da Tabela e ou View em questão. Uma Trigger não pode ser disparada pela sua aplicação como é feito com a SP, trigger só é disparada pelas ações de INSERT/UPDATE/DELETE. No Firebird existem dois tipos de Trigger:

BEFORE;
AFTER;

Isto é, é disparado um gatilho antes do INSERT/UPDATE/DELETE e ou é disparado um gatilho após o INSERT/UPDATE/DELETE.

Para maiores detalhes da sintaxe da criação de SP's e ou Trigger, consulte os manuais do Firebird.

Eventos do Firebird:

O mecanismo de envio dos eventos consiste em três partes:

- A) A Trigger e ou a SP posta o evento para o gerenciador de eventos do Firebird;
- B) O evento entra em uma fila de Eventos gerenciada pelo Gerenciador de Eventos e notifica as estações que ocorreu um evento;
- C) A Aplicação/Estação registra ou não o interesse do Evento. Isto é, é a aplicação que toma a decisão ou não de registrar o evento. A aplicação não tem o controle de dizer ao Firebird para o mesmo não postar o evento!*

*Existem dois artifícios da aplicação para dizer ao Firebird que não deseja receber tal evento. O 1º é de não registrar o evento e será explicado mais adiante, o 2º só funciona com SP, onde você pode passar um parâmetro para a Stored Procedure indicando que não deseja que o Firebird poste tal evento!

POST_EVENT <nome_do_evento>;

É este comando responsável por notificar as estações que ocorreu um evento no Banco de Dados. O nome_do_evento é Case-Sensitive, isto é, se você declarou um evento com o nome de:

```
POST_EVENT 'Inclusão_Cliente';
```

A sua aplicação tem que fazer referencia ao nome exatamente como está descrito no POST_EVENT. Até a versão 6.0 do Firebird e o FireBird, o tamanho máximo do nome do evento é de 64 caracteres.

Vejamos um exemplo de utilização do POST_EVENT dentro de uma Stored Procedure.

Nota: No Banco de Dados de Exemplo que acompanha o FireBird, existe um evento declarado na tabela SALES para a Trigger AFTER INSERT. Como mostra a nossa tela:

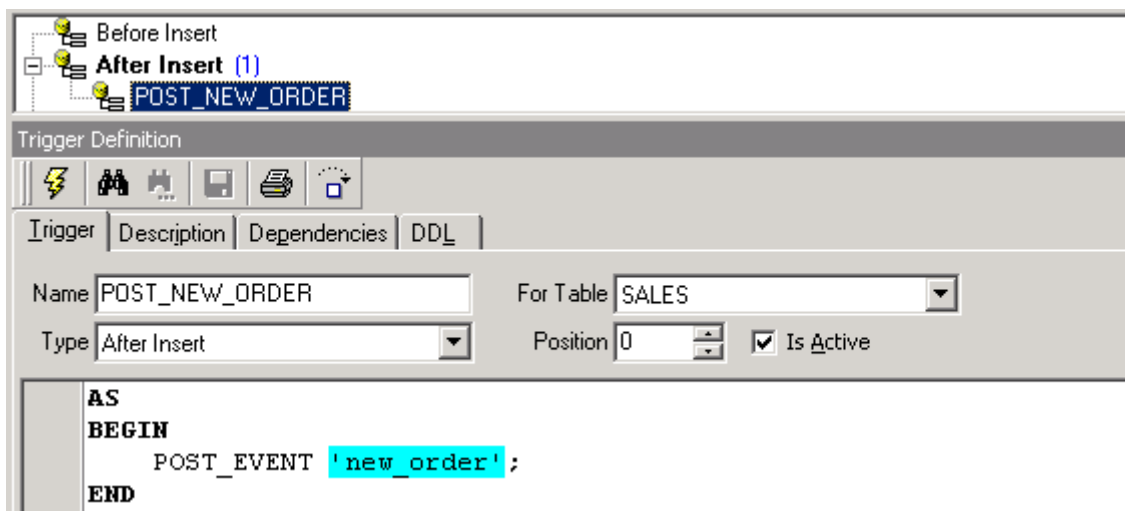


Figura 1: A declaração do Evento na Tabela SALES.

O evento tem o nome de: 'new_order' e POST_NEW_ORDER é o nome da Trigger.

Para nosso artigo, irei mostrar como usar em uma SP o recurso de Eventos do Firebird.

Criei um Banco de Dados Firebird com o seguinte nome: EVENTOS.GDB

Neste Banco de Dados Firebird, eu criei uma SP com o seguinte nome: EVENTOSDEMO, e na declaração da SP iremos encontrar o seguinte código SQL:

```

SET TERM ^;
CREATE PROCEDURE EVENTOSDEMO (EVENTO VARCHAR(64))
AS
BEGIN

    POST_EVENT :evento;

END^
SET TERM^;

```

Ok! Nosso Banco de Dados de Eventos criado e nele vamos encontrar a nossa SP que será responsável em disparar o evento!

Vamos criar nossa aplicação Demo de Eventos no Delphi. Abra o Delphi, e inicie uma nova aplicação. Nosso projeto se chamará: EventosIB. Para o nosso projeto, eu utilizei os componentes da ABA Interbase do Delphi, estes componentes são chamados de InterBase Express (vulgo IBX), a versão do IBX usada neste exemplo é: 6.02, mas, deve funcionar com uma versão menor. Caso queira atualizar o seu IBX, vá até o seguinte site da Internet: <http://codecentral.borland.com/codecentral/ccweb.exe/author?authorid=102>

Vamos adicionar ao nosso projeto um DataModule “DM” que terá os seguintes componentes:

- IBDataBase com a propriedade Name= IBDatabaseEventos;
- IBTransaction com a propriedade Name= IBTransactionEventos;
- IBStoredProc com a propriedade Name= StoredProcEventos;
- IBEvents com a propriedade Name= IBEventoAlerta;



Figura 2: O DataModule da Aplicação.

Temos também no nosso projeto um formulário com o nome uEvento que contém os seguintes componentes:

- Dois TGroupBox;
- Um TLabel
- Um TEdit;
- Um TButton;
- Um TListBox;

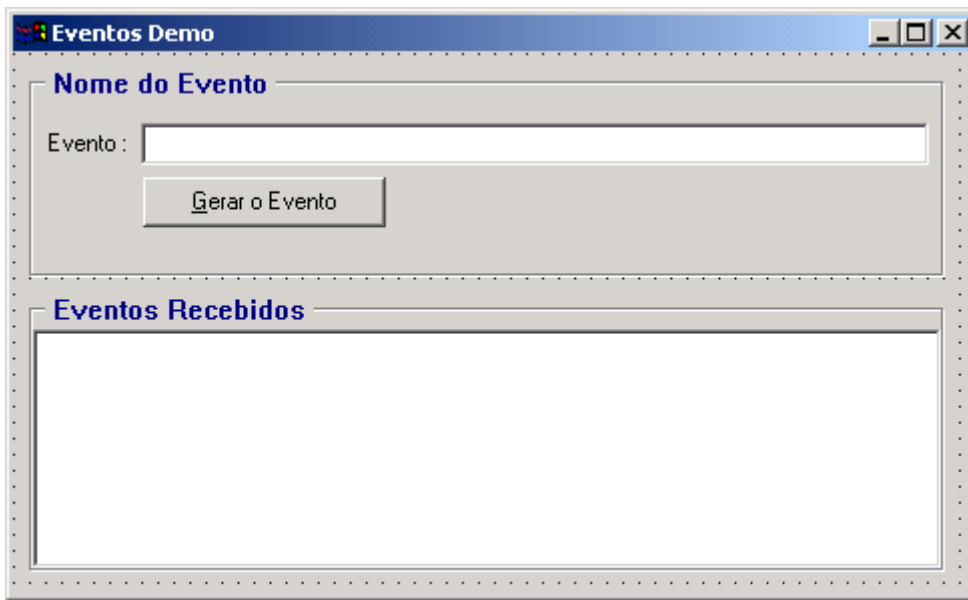


Figura 3: O Formulário da Aplicação.

Ok. Já temos a interface do nosso projeto pronta!

Voltamos ao DataModule, vamos indicar ao IBDataBase onde está o nosso “EVENTOS.GDB”. Duplo clique no componente IBDataBase e irá abrir o DataBase Editor conforme figura 4:

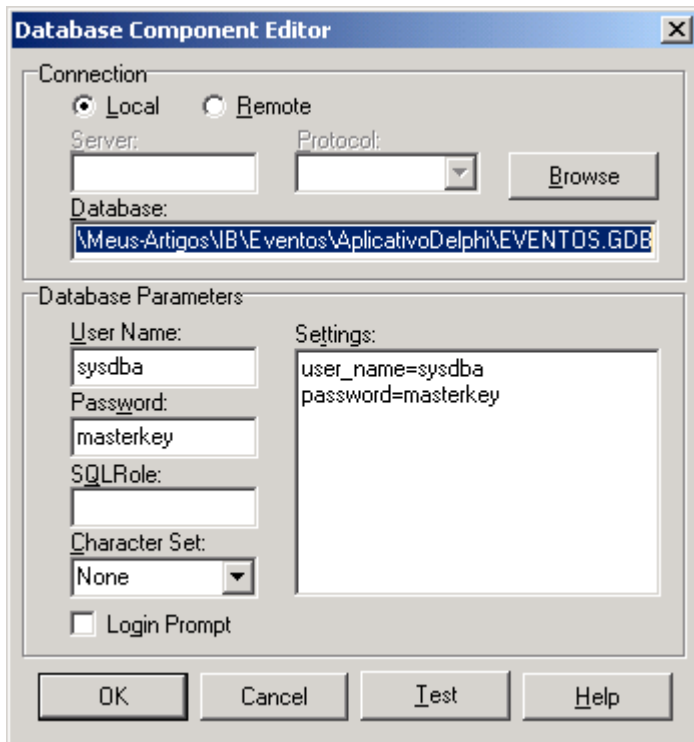


Figura 4: DataBase Editor do Componente IBDataBase.

Em Connection, escolha a opção Local, em DataBase especifique o caminho onde se encontra o seu EVENTOS.GDB, ex:

D:\\Anderson\\Meus-Artigos\\IB\\Eventos\\AplicativoDelphi\\EVENTOS.GDB

User Name coloque SYSDBA, Password=masterkey, SQL_Role deixe em branco, Character Set=None, e desmarque a opção Login Prompt. Ok. Temos a nossa tela 4 e já criamos a nossa conexão com o Banco de Dados EVENTOS.GDB.

No DataModule ainda, no IBDataBase indique na propriedade DefaultTransaction o nosso IBTransaction, no IBTransaction na propriedade DefaultDataBase indique o nosso IBDataBase, no IBStoredProc na propriedade DataBase indique o nosso IBDataBase e em Transaction indique o nosso IBTransaction, no IBEvent indique na propriedade DataBase nosso IBDataBase! Ufa! Pronto, criamos a nossa conexão! Se tudo estiver certo, faça o nosso IBDataBase se conectar com o Banco através da propriedade Connected, logo após, não esqueça de retornar para False novamente!

Arquivo de Configuração para receber os eventos

Como foi explicado mais acima no artigo, o Firebird dispara o evento e a sua aplicação é que a responsável em decidir se irá ou não registrar o evento disparado pelo Banco. A saída para este caso, é termos um arquivo texto para cada estação indicando quais os eventos do Banco de Dados cada estação pode e vai receber*.

No nosso exemplo, junto com o executável existe um arquivo chamado EventosEstacao.cfg e neste arquivo texto existe o nome de cada evento “cadastrado” no Banco de Dados que a estação vai registrar! Segue um exemplo de como está no nosso exemplo:



Figura 5: Nosso arquivo de Configuração para cada estação.

Ok. Vamos agora para a parte de codificação do projeto.

No evento OnCreate do nosso DataModule coloque o seguinte código:

```
procedure TDMEventos.DataModuleCreate(Sender: TObject);
var ArqEventos: TStringList;
begin
  IBDatabaseEventos.Open;
  ArqEventos := TStringList.Create;
  try
    { Abre o arquivo de configuração dos registros. }
    ArqEventos.LoadFromFile('EventosEstacao.cfg');
    with DMEventos.IBEventoAlerta do
    begin
      UnregisterEvents;
      Events.Text := ArqEventos.Text;
      { Registra os eventos! }
      RegisterEvents;
    end;
  finally
    ArqEventos.Free;
  end;
end;
```

É no código mostrado acima que a nossa estação registra os eventos que ela pode receber. A “mágica” está nos métodos UnregisterEvents e RegisterEvents, onde, cada um é responsável pela ação de registrar os eventos e de não registrar/aceitar os eventos!

Ainda no DataModule, precisamos programar mais dois eventos:

O OnDestroy do DataModule e o OnEventAlert do IBEvents:

```

procedure TDMEventos.DataModuleDestroy(Sender: TObject);
begin
  if IBDatabaseEventos.Connected then
  begin
    { Se a transação está aberta, precisamos fechá-la }
    if IBTransactionEventos.InTransaction then
      IBTransactionEventos.Commit;
    { Se existe o registro de eventos, precisamos desalocar estes recursos }
    if IBEventoAlerta.Registered then
      IBEventoAlerta.UnRegisterEvents;
    { Fecha a conexão com o Banco de Dados }
    IBDatabaseEventos.Close;
  end;
end;

```

O código já está comentado, desta forma, acredito que não preciso explicar o mesmo!

```

procedure TDMEventos.IBEventoAlertaEventAlert(Sender: TObject; EventName: string;
  EventCount: Longint; var CancelAlerts: Boolean);
begin
  { Adiciona a linha com os nomes do eventos que foram disparados pelo Botão “Gerar o Evento” do Formulário }
  frmEventos.LstBxRecebidos.Items.Add(EventName);
end;

```

Veja que o DataModule e o Formulário estão “ligados” entre si através da cláusula uses da implementation de cada um!

Finalizamos a programação do nosso DataModule. Vamos agora programar o nosso formulário. Nosso formulário tem apenas um evento, é o evento OnClick do botão “Gerar o Evento”. Veja o código:

```

procedure TfrmEventos.btnGerarEventoClick(Sender: TObject);
begin
  with DMEventos do
  begin
    { Iniciamos a transação. }
    IBTransactionEventos.StartTransaction;
    StoredProcEventos.Prepare;
    { O nome do evento digitado no TEdit. }
    StoredProcEventos.Params[0].AsString := EdtEvento.Text;
    StoredProcEventos.ExecProc;
    { Fechamos a transação. }
    IBTransactionEventos.Commit;
  end;
end;

```

Nosso código é simples e já está comentado! Mas, vou citar que se você digitar um nome de evento que não exista no arquivo de configuração, não será adicionado nada no nosso ListBox, por que o mesmo não foi registrado no arquivo de configuração, mas, ele foi gerado pelo Firebird e está na fila de eventos registrados pelo Banco Firebird!

Vale mencionar novamente, que os nomes de Eventos são Case-Sensitive, então se você digitar algum caracter, o mesmo não será adicionado ao nosso ListBox!

A nossa tela com os três eventos registrados, disparados e resgatados pela aplicação:

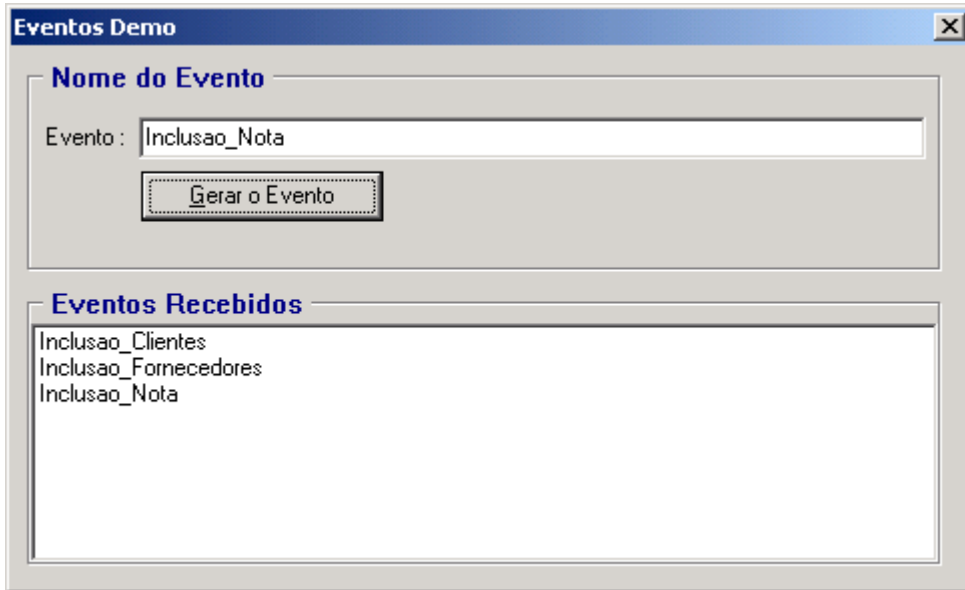


Figura 6: A nossa aplicação final com os eventos registrados e resgatados.

Alguns métodos disponíveis em nosso IBEvents;

Propriedades:

AutoRegister: Registra automaticamente os eventos cadastrados em Events quando houver a conexão com o Banco de Dados Firebird.

Events: Nome dos eventos cadastrados no Banco de Dados.

Registered: Indica que IBEvents registrou todos os eventos, tem a mesma função do método RegisterEvents.

Eventos :

OnEventAlert: Ocorre quando algum evento é recebido pelo Objeto.

OnError: Ocorre quando houver algum erro na captura do Evento.

Principais Métodos:

QueueEvents: Indica a sua aplicação, que foi inicializada o recebimento de eventos.

RegisterEvents: Registra ao IBDataBase, os eventos listados em IBEvent's.

UnRegisterEvents: Diz ao Banco de Firebird, que está retirando os registros do IBEvent's.

Conclusão:


Eventos é um poderoso recurso que o Banco de Dados Firebird nos oferece, podemos através dele saber quando houve uma mudança em uma Tabela do Banco de Dados. Com os eventos, podemos resolver o “problema”** de refresh de nossas estações em relação a tabela do Banco de Dados, onde, podemos programar o evento para o mesmo fechar a nossa tabela e abrir novamente, sendo assim, temos exatamente em nossa tela o novo registro incluído/alterado e se for o caso deletado por outra estação!

*Nota1: Foi citado que a única saída para a configuração dos eventos em cada estação, é termos um arquivo de configuração para cada estação! Citei esta forma por ser a mais fácil e rápida de ser feita, mas, acredito que a melhor maneira para resolver este problema de cada estação ter o seu arquivo é o mesmo arquivo de configuração apenas no servidor. Mas, a estrutura do arquivo de configuração mudaria, e neste caso podemos usar a mesma estrutura de um arquivo .INI, onde o cabeçalho de cada seção pode ser o nome da estação e ou o IP da máquina. Podemos então acessar cada seção através da classe do Delphi TIniFile que está declarada na unit IniFiles!

**Nota2: Foi citado também um problema de refresh entre estações, para as estações enxergarem o que as outras estão fazendo, é necessário no seu aplicativo que a transação em questão esteja com a opção “ReadCommitted”, isto é, enxergar as mudanças das estações em relação ao estado atual do seu Banco de Dados.

Sucesso a todos,

Anderson Haertel Rodrigues (AHR) é administrador de bancos de dados, desenvolvedor de sistemas e componentes. Mora em Florianópolis/SC e pode ser contactado em anderson.hr@bol.com.br.

<p>Artigo Original</p> <p>Anderson Haertel Rodrigues (Colaborador da CFLP)</p> <p>anderson.hr@bol.com.br</p>	
	<p>Comunidade Firebird de Língua Portuguesa</p> <p>Visite a Comunidade em:</p> <p>http://www.comunidade-firebird.org</p>
<p>A Comunidade Firebird de Língua Portuguesa foi autorizada pelo Autor do Original para divulgar este trabalho</p>	