Autor: **Amarildo Lacerda**
Objetivo: **Consolidar publicações do Facebook em um documento base**

Delphi

Delphi - Trabalhando com Class Helper

#delphi #helper #fevereiro/2016

A introdução de Class Helper no delphi ajudou o desenvolvedor a incluir novas funcionalidades nas classes nativas do delphi – você tem uma idéia de um recursos dos sonhos e gostaria de implementar lá na classe base, use Class Helper

Exemplo, extendendo o TDataset:

```
TDatasetHelper = class helper for TDataset
 public
     procedure Run(AProc:TProc<TDataset>);
end;

procedure TDatasetHelper.Run(AProc: TProc<TDataset>);
begin // executa em um thread separada
   TThread.CreateAnonymousThread(
   procedure begin
       if assigned(AProc) then
           AProc(self);
       end
   ).Start;
end;
```

(fonte: https://github.com/amarildolacerda/helpers/blob/master/Data.DB.Helper.pas)

O exemplo implementa a capacidade de uso da "procedure" RUN(...) por qualquer classe com herança de TDataset.

Trabalhando com o novo procedimento:

O procedimento RUN por finalidade executar em um processo paralelo (uma nova TThread) a rotina que estiver implementando no TProc<TDataset>.

```
var total:double;
begin
        query1.run(
            procedure (ds:Tdataset)    // rotina a executar em paralelo.
            begin
                ds.first;
```

```
                    while ds.eof=false do
                    begin
                          total := total + xxxxx;
                          ds.next;
                    end;
              end);


end;
```

Limitações:

- o uso de Helpers esta limitado a um por unit para a mesma classe;
- dificuldade em lidar com variáveis novas locais ao helper;

Delphi – Trabalhando com RECORD Helper

Assim como Class Helper extende classes, Record Helper permite extender Records.

Quantas vezes já precisou fazer: total := total + x;

Utilizando helper poderia escrever assim: total.add(x);

(fonte: https://github.com/amarildolacerda/helpers/blob/master/System.SysUtils.Helpers.pas)

Delphi - Computação Paralela, em Threads anônimas

#delphi #parallel #thread #janeiro/2016

Houve um tempo que executar tarefas em "background" era um tortura.
Com a evolução de hardware e software, fazer uso de computação paralela passou a ser tão fácil quanto um desafio em resistir ao seu uso.

Introduzida na versão XE7, um "thread" passou a rodar considerando o balanceamento entre os processadores presentes na unidade de processamento.

Associar o uso de métodos anônimos facilitou em muito todo o trabalho:

```
Tthread.CreateAnonymousThread(

procedure begin

   // execute aqui o seu código

  // se precisar sincronizar atualização de controles da janela

  TThread.Queue(nil,

  procedure begin

     // aqui deve incluir a atualização dos controles da janela;

    // ex: label1.caption := 'texto';

end);

end

).Start;
```

#firebird #delphi #sql

Delphi - Processamento paralelo com TTask #task #parallel #thread

Na versão XE7 foi incorporado ao Delphi o conceito de processamento paralelo. Antes já era possível fazer algo parecido utilizando bibliotecas de terceiros [http://www.omnithreadlibrary.com/index.htm http://andy.jgknet.de/blog/bugfix-units/asynccalls-

29-asynchronous-function-calls/].
Para tentar simplificar o conceito, diz paralelo quanto consegue executar dois ou mais processos ao mesmo tempo, daí o nome "paralelo".

Exemplo utilizando TTask:

```
var
  tsk: array [0 .. 2] of ITask;
  i, n: integer;
begin
  tsk[0] := TTask.create(
    procedure
    begin
     TThread.Queue(nil, procedure
     begin
      caption := 'xxx';   // sincronizar a atualização da janela.
     end);
    end);
  tsk[0].Start;  // inicia o processamento antes de criar as TTask seguintes

  tsk[2] := TTask.create(
    procedure
    var
      k: integer;
    begin
      i := 1;
      sleep(1000);
      for k := 0 to 10000 do
        inc(i);
    end);

  tsk[1] := TTask.create(
    procedure
    var
      k: integer;
    begin
      n := n;
      for k := 0 to 1000 do
        inc(n);
      add( 'N');
    end);

  tsk[2].Start;   // inicia o processamento
  tsk[1].Start;

  TTask.WaitforAll(tsk);  // quando quer aguardar o termino de todas as TTasks
```

Diferença entre #ITASK e #IFUTURE. ..

interfaces disponíveis através de TTASK que se encontra na unit System.Threading...

Enquanto ITask aguarda uma chamada x.start; para iniciar o processamento em uma thread própria... o IFuture já inicia o processamento de imediato e aceita aguardar uma resposta após o término de execução.

Exemplo ITask: (aguarda start para iniciar)

```
tsk := TTask.create( procedure begin
            // código a ser executado
       end);
tsk.start;

//pode ou não pedir para aguardar conclusão:
  tsk.wait( 1000000);
```

Exemplo IFuture: (inicia assim que for criado)

```
tsk = TTask.Future<boolean>(
          function:boolean
          begin
               // código.....
               result := true;   // resposta para o processamento
          end);

resposta :=  tsk.value;       // aguarda a thread terminal e pega o resultado
```

Se precisar retornar mais de um valor, pode usar um RECORD:

```
TPessoa = record
   nome:string;
   idade:integer;
 end;

tsk = TTask.Future<TPessoa>(
          function:TPessoa
          begin
               // código.....
               result.nome := 'Nome';   // resposta para o processamento
               result.idade := 18;
          end);

resposta :=  tsk.value;       // aguarda a thread terminar e pega o resultado

if resposta.idade>18 then
   xxxxx
```

ver: #ttask

Delphi - Executando uma Query em Segundo

Plano.

#delphi #firedac #query #tthread #parallel

Executar uma query em segundo plano (em paralelo) não ...é difícil de fazer, o seu controle é que pode ser mais complexo.

Para executar em segundo plano basta:

```
 TThread.CreateAnonymousThread(procedure
begin
      ALQuery1.sql.Text := '....';
      ALQuery1.Open;
 end).Start;
```

ou

```
Tthread.CreateAnonymousThread(
```

```
  procedure
      var i:integer;
      begin
        for I := 0 to 10 do
        begin
            // faz alguma coisa...
            AlQuery1.execSQL;
        end;
    end).Start;
```

Um pensamento simplista é você criar uma conexão para cada QUERY em separado. Se você tem um CONNECTION isolado para UMA QUERY, então é possível executá-la em paralelo dentro de uma nova Thread;

Algumas idéias onde pode utilizar o processo em paralelo:
- quando precisa registrar um log em uma tabela;
- se for possível adiantar um SELECT que será utilizado mais a frente;
- se precisa rodar um loop que não tem dependência com os próximos passos da sequencia do código;
- quando precisa fazer um somatório de dados na tabela para mostrar o seu valor na janela.... e liberar o usuário para continuar fazendo outras coisas.

As vezes você pode por um SELECT em paralelo e disparar outra sequencia de código... e mais adiante aguardar o primeiro SELECT concluir... para depois então continuar.... Este é assunto para outro POST para tratar de TTASK.

Delphi: Dataset Loop - Anonimous Method

#delphi #firedac #dataset #anonimous

Lembra quantas vezes você precisou fazer um Loop em um Dataset para fazer uma soma, uma contagem ou qualquer outra coisa...
Não gosto de fazer de novo algo que já fiz antes... Pensando nisto passei a usar "anonimous method" do delphi para executar para mim os trechos repetitivos dos loops...
Veja como ficou.

```
type
    TDatasetHelper = class helper for TDataset
    public
       procedure DoLoopEvent(AEvent: TProc<TDataset>); overload;
    end;

procedure TForm34.execute;
var total:Double;
begin
     // abere o Dataset com os dados.
     alQuery1.sql.Text := 'select codigo, total valor from sigcaut1 where data>=:data';
     alQuery1.ParamByName('data').AsDateTime := strTodate('01/01/2016');
     alQuery1.Open;
     // fazer um loop para somar o total, usando metodos anonimos;
     total := 0;
     alQuery1.DoLoopEvent( procedure( ds:TDataset)
        begin
          total := total + ds.FieldByName('valor').AsFloat; // executa o loop
        end);
    showMessage( FloatTOStr(total) ); // mostra o total da soma obtida no loop
end;

procedure TDatasetHelper.DoLoopEvent(AEvent: TProc<TDataset>);
var
  book: TBookMark;
begin
  book := GetBookmark;
  try
     DisableControls;
    first;
    while eof = false do
    begin
        AEvent(self);
       next;
    end;
  finally
     GotoBookmark(book);
     FreeBookmark(book);
     EnableControls;
  end;
end;
```

Código Original: https://github.com/amarildolacerda/...

Obter Json de uma classe genérica

#delphi #json

Requer: https://github.com/amarildolacerda/helpers

```
Uses System.uJson;

type
  TMinhaClasse = class
  public
    Valor: Double;
    Codigo: string;
  end;

procedure TForm33.FormCreate(Sender: TObject);
var
  mc: TMinhaClasse;
begin
  mc := TMinhaClasse.create;
  try
    mc.Codigo := '123456';
    mc.Valor := 10;
    ShowMessage(mc.asJson);
  finally
    mc.Free;
  end;
end;
```

Firebird

#Firebird - #Backup #Incremental

O Firebird traz uma ferramenta de backup pouco falada - #nbackup - que faz backup incremental. Quando roda o backup deve indicar qual o nível deseja... Nível 0 - backup completo. Nível 1 - somente a diferença entre o último nível 0 e o agora. Nível 2 - somente a diferença desde o último nível..... Importante - ele não faz validação dos dados precisa de estratégia em paralelo para validação - é muito bom para fazer cópia sem parar o banco "a quente". É muito rápido, bom para fazer cópias intermediárias durante a operação.

Exemplo Nivel 0:

```
nbackup -B 0 localhost:c:\dados\meubanco.fdb c:\backup\backup.bak -U sysdba -P masterkey
```

Exemplo Nivel 1:

```
nbackup -B 1 localhost:c:\dados\meubanco.fdb c:\backup\backup1.bak -U sysdba -P masterkey
```

Como restaurar Nivel 0 +Nivel 1:

```
nbackup -R c:\dados\teste.fdb c:\backup\backup.bak c:\backup\backup1.bak
```

Interessante: permite fazer o backup de uma máquina remota.

nbackup com Firedac #firedac #nbackup #firebird

A API do firedac traz um componente que encapsula o nbackup do firebird o que facilita personalizar o controle de backups. TFDFBNBackup.

Exemplo Nivel 1:

```
TNBackup.ExecuteNBackup('localhost','c:\dados\meubanco.fdb','sysdba','masterkey',1,'c:\backup\backup2.nbk');
```

Segestão de como utilizar NIVEL (level):
a) fazer backup FULL Nivel 0 para um intervalo de período (semanal);
b) fazer backup Nivel 1, diário;
c) fazer backup Nivel 2 para backup a cada hora.

Codigo base:

```
uses
FireDAC.Phys.IBWrapper,FireDAC.Phys.FB,FireDAC.Phys.FBDef,FireDAC.Comp.UI,FireDAC.Phys;

type
  TNBackup = record
  private
  class function GetNBackup(AHost, ABanco, AUser, APass: string;
```

```
     ANivel: integer; ADestino: String): TFDFBNBackup;static;
    class function ExecuteNBackup(AHost, ABanco, AUser, APass: string;
     ANivel: integer; ADestino: String): boolean;static;
    end;

class function TNBackup.ExecuteNBackup(AHost, ABanco, AUser, APass: string;
 ANivel: integer; ADestino: String): boolean;
begin
  result := false;
  with TNBackup.GetNBackup(Ahost,ABanco,AUser,APass,ANivel,ADestino) do
  try
    Backup;  // gerar backup.
    result := true;
  finally
    free;
  end;
end;

class function TNBackup.GetNBackup(AHost, ABanco, AUser, APass: string;
 ANivel: integer; ADestino: String): TFDFBNBackup;
var
   nBackup:TFDFBNBackup;
   FDGUIxWaitCursorX: TFDGUIxWaitCursor;
   FDPhysFBDriverLinkX: TFDPhysFBDriverLink;

begin
   result:=TFDFBNBackup.create(nil);
   try

   FDGUIxWaitCursorX:= TFDGUIxWaitCursor.Create(result);
   FDPhysFBDriverLinkX:= TFDPhysFBDriverLink.Create(result);

      with result do
      begin
        Level := ANivel;
        host := AHost;
        username := AUser;
        password := APass;
        protocol := ipTCPIP;
        Database := ABanco;
        backupfile := ADestino;
        DriverLink := FDPhysFBDriverLinkX;
      end;
   finally
     // liberar a instancia no metodo chamador
   end;
end;
```

Firebird - CTE "Common Table Expression"

#firebird #delphi #sql

Complexas consultas em Firebird podem ser resolvidas utilizando CTE.

CTE é uma construção que monta uma tabela de memória a ser utilizada em um SELECT e é desmontado quando termina a execução do SELECT.

1) Considerando uma tabela de vendas (existente no banco de dados):

```
CREATE TABLE vendas (codigo varchar(20), data date, qtde numeric(18,4), valor numeric(18,4) );
```

2) Inserir alguns dados na tabela que simule as vendas;

3) Executar a seguinte consulta no banco de dados:

```
with VendasDoMes as
      ( -- agrupa os dados de vendas
          select Codigo,
             extract(month from data) Mes,
             extract(year from data) Ano,
             sum(qtde) Qtde,
             sum(valor) Total
          from vendas
          group by 1,2,3 -- agrupa as colunas
      )
select * from VendasDoMes --monta a consulta
 where ano=2015 and mes=12
 order by mes, ano
```

4) Conclusão:
O SELECT irá retornar a quantidade e total de vendas de todos os produto no mês de Dezembro de 2015.

Teste unitário em Firebird !!!

As linguagens de programação em geral evoluíram nos últimos anos em direção à qualidade.
E os banco de dados ?

Ainda que não seja popular, com um pouco de imaginação é possível fazer teste unitário em banco de dados firebird.

Passos para fazer em firebird:
1) criar um modelo de exception para retornar informações;

```
create or alter exception test_Error 'Error: ';
```

2) montar um esqueleto da procedure, função ou package a ser testada;

```
SET TERM ^ ;
CREATE OR ALTER PROCEDURE DECODEDATE (data date)
   returns (ano integer,mes integer,dia integer)
as
begin
    DIA = EXTRACT( DAY FROM DATA);
    MES = EXTRACT( MONTH FROM DATA);
    ANO = EXTRACT( YEAR FROM DATA);

    suspend;
end^
SET TERM ; ^
```

3) **usar Block para executar o teste e se falhar retornar mensagem de erro**;

```
SET TERM ^ ;
execute block as
declare variable dt date;
declare variable y double precision;
declare variable m double precision ;
declare variable d double precision ;
begin
  -- realizar o teste
  dt = cast('now' as date);
  select y,m,d from DecodeDate( :dt )
  into :y,:m,:d;
  if (:y <> extract(year from :dt)) then
     exception test_Error 'DecodeDate Year';
  if (:m <> extract(month from :dt)) then
     exception test_Error 'DecodeDate Month';
  if (:d <> extract(day from :dt)) then
     exception test_Error 'DecodeDate Day';
end^
SET TERM ; ^
```

Ver exemplo completo: https://github.com/…/08.03.002_firebird_packages_dateutils.…

Firebird - Trabalhando com 1.000.000 de itens cadastrados

#brincandoComFirebird

Na escola, meus amigos gostavam de jogar vídeo games, eu não sei nada - o que me fascinava mesmo era como eram feitos - até "assembly" foi aprender por conta disto.
Depois fiquei fascinado com um tal de Lotus 123. O Clipper, há ninguém lembra mais, era o bicho. Já usava scripts (nem existia js)... Mas tudo mudou mesmo com Delphi 1.0... É ai veio Delphi+Firebird, não quero mais sair.
O lance agora é brincar de firebird com delphi... video game - só usa o que já existe, pronto, sem criatividade do participante.

Para brincar de Firebird, queria fazer um desafio - quero por 1 milhão de itens no cadastro de produto e fazer SELECT para localizar um produto usando só uma parte do nome (com performance... por favor).

As fases da brincadeira seria assim:
a) criar tabela para receber os nomes, vamos por: codigo,nome,unidade,preco; (não importa o conteúdo, só para preencher a tabela mesmo) - mas tem uma regra, não pode ficar repetindo os mesmos nomes - para não perder a graça.
b) ai vou precisar de ajuda, vamos montar scripts para preencher esta tabela até ela chegar a 1.000.000 de linhas; Ela terá somente um índice (nome);

c) depois que estivermos com os scripts carregados, cada um vai montando o seu banco de dados com eles.... aí nós vamos construir o SELECT para fazer a busca.

d) la nesta lista vamos ter uma em particular:
"Boneca lindinha azul 2016 para 5 anos"

e) o SELECT... deve retornar somente os produtos que contenham o nome "lindinha" + "2016"

Vou subir os primeiros 100.000 produtos no GIT, depois passo o link... com uma lista contendo os inserts... quem tiver alguns podem mandar... até completar 1.000.000;

Quem quiser participar - deixa aí a mensagem... e vamos brincar um pouco de firebird.

A Tabela:

```
Create table FB_PROD (
    codigo varchar(18),
    nome varchar(50),
    unidade Varchar(5),
    preco Double Precision);

create index FB_PROD_NOME on FB_PROD(nome);
```

por enquanto é só (podem mandar os inserts no meu endereço...)

Firebird - Nem sentiu 1.000.000 de produtos - Veja como melhorar

#brincandoComFirebird

Completado o objetivo de coletar 1.000.000 de linhas em um cadastro de produtos como desafio cumprido podemos colher algumas conclusões:

Com esta quantidade de registros em uma tabela que contém somente 4 colunas, o desempenho do SELECT:

a) 1.327 seg - usando "like" que não usa índice '%lindinha%' (no meio do nome)
b) 1.453 seg - se fizer "like" para buscar um nome que não existe no banco de dados
c) 0,047 seg - quando usa like que busca pelo índice 'Boneca%' (no início do nome)

Acreditando que estes números podem representar maior impacto em produção considerando a associação Quantidade de Acessso x Numero de Colunas na Tabela, então vamos reestruturar a estratégia de busca.

Observando o resultado "c" nota-se que o motor utilizou o índice por nome. Então o que precisa ser feito é conseguir alterar a estratégia de tal forma que retorne as mesmas linhas de "a", mas utilizando índice para otimizar a busca.

Restruturando:
1) criar ums nova tabela que recebe palavras chaves do nome e associe com o código do produto;

2) criar índice para a nova tabela (um para cada coluna) código, palavra;
3) criar trigger para fazer a atualização da tabela de palavra chave toda vez que o usuário inserir, apagar ou alterar um nome na tabela de produtos;

Feito a mudança na estratégia de busca na tabela, gostaria de melhorar o resultado na situação “a” para que retorne em menor tempo - mais próximo ao registrado em "c". (fontes em: https://github.com/amarildolacerda/brincandoComFirebird)

Reestruturado os dados de forma que fosse possível fazer o SELECT (que retorna as mesmas linhas do exemplo “a”):

```
select codigo,nome
from fb_prod a, palavra_chave_fb_prod b
WHERE
   a.codigo=b.CODIGO
   and
   b.palavra like 'lindinha%'
   AND
   b.palavra like '2016%'
```

Lógica:
- usuário insere um registro novo;
- a trigger usa a procedure que fazer a atualização da tabela de palavras chave;
- a procedure de atualização usa um SPLITTER para quebrar o nome do produto em palavras chaves e guarda o resultado no dicionário de palavras chave (que agora permite índice de otimização) Os fontes estão no GIT para baixar.
https://github.com/amarildolacerda/brincandoComFirebird

Faça os testes ai na sua base de dados e note como vai melhora muito o resultado da pesquisa.

Post Inicial: https://www.facebook.com/groups/DelphiExperts/1740425566189192

Firebird - Utilizando funções financeiras da PKG_FINANCE

(package do Firebird 3.0) para calcular o valor de uma prestação constante:

onde:
Valor do Financiamento = 30000,00
Taxa de Juros = 5% am
Parcelas = 36

```
select PKG_FINANCE.pmt(30000,5,36) FROM RDB$DATABASE
```

Resposta: 1813,03

Para baixar o código: https://github.com/amarildolacerda/firebird

Firebird Packages DateUtils Like

#firebird #packages #delphi #sql
Um dia desses estava estudando o Firebird 3.0 e para deixar a brincadeira divertida passei a escrever uma “Package” para entender o poder do recurso. Fim da graça resultou na biblioteca PKG_DateUtils, uma tentativa de imitar alguns recursos existente no Delphi na UNIT de mesmo nome.

Ver o código: https://github.com/amarildolacerda/...

Como utilizar:
Tomando por base que a maior parte é publicada como funções ....

```
select pkg_dateutils.dayOfTheWeek('today') from rdb$database
```

Para ver uma lista completa fazer leitura dos fontes onde (no final) consta a rotina de teste da “Package”;

Chutar o Banco de Dados e depois reclamar não é racional

#firebird
Nos últimos meses dedico algum tempo lendo as mensagens postadas nos foros e a tempo estava me coçando para dar meu palpite, dado aos erros básicos que vejo - que vou tentar lembrar de alguns.

1) fazer um select de todas as linhas da tabela e depois fazer um filtro para pegar um único registro. Correto: é construir um select que retorna só registros de interesse; Ex: `select codigo,nome from usuario where codigo=’123’`
2) fazer pouca gestão sobre índices é um erro; Correto: é você por no banco de dados somente índices que sejam favoráveis... lembrando, um índice demanda tempo de processamento quando faz INSERT, UPDATE ou DELETE; O SELECT demanda tempo para o motor do banco de dados escolher qual índice é melhor (no prepare); Então não crie muitos índices se você executa muitas atualizações... No Update, não altere colunas que não sofreram alteração, para não demandar tempo para atualizar o índice (altere somente o necessário);

3) colunas que são candidatas a índice são aquelas que fazem parte de uma WHERE, ORDEM BY ou GROUP BY... nem todos devem ser transformados em índices... somente alguns... entenda como o banco de dados usa as estatística para escolher qual o índice é mais adequado para inclui no plano de otimização;

4) índice composto ou simples ? depende do banco de dados... no Firebird sempre que testei o resultado melhor foi obtido em índices simples Ex: create index cadastroNome on produtos(codigo,nome)..... pode não ser tão eficiente quanto criar 2 índices... analise o resultado antes de criá-los;

5) os índices são bons se a coluna registrar o maior numero possível de informações diferentes... por isto o melhor índice é o da chave primária, pois existe somente uma linha possível, nenhuma é igual a outra. Um índice de FILIAL em uma tabela pode não valer NADA se a empresa tem uma única

loja... o mesmo vale para uma coluna S ou N - resulta em muitas linhas com o mesmo valor;

6) usar funções dentro de WHERE ou GROUP BY deve ser avaliado com cuidado, na dúvida evite-os.

7) não use NOT IN ou <> em uma WHERE, em geral o gerenciador ignora todos os índices nestas situações... Ex: Errado: select qtde from vendas where data<>'today' (isto mata o índice)... prefira Correto: select qtde from vendas where 'today' >data and 'today' < data (isto vai permitir usar o índice de data);

8) não tenha preguiça.... não pode fazer "select * from xxxxx" (digite as colunas que vai precisar), isto vai economizar tempo do servidor e de banda de transmissão dos dados;

9) se a sua tabela tem menos de 1000 linhas.... é provável que nem precise de índice. Mas se ela tem 1 milhão de linhas, não dá para ignorar o índice;

10) evite até a morte em criar uma tabela sem indicar a sua chave primária; Bancos de dados foram feitos para ter chave primária na mais profundo da sua concepção. De a ele uma chave primária para que ele fique feliz;

11) sempre que for fazer um SELECT tente minimizar ao máximo o número de registro que ele irá buscar no banco de dados... Se vai precisa da QTDE vendida e o NOME do produto, não faça 2 SELECTs.... junte os 2 em um único SELECT;

12) se acabou de buscar os dados de um cadastro.... não vai buscar de novo o que já conhece... guarda na memória e reutiliza a informação que já foi obtida. Por isto o GOOGLE é tão rápido - ele memoriza tudo e vai no buscar somente dados que ainda não conhece.

13) No FIREBIRD, deixar conexões sem dar commit ou as vezes em aberto sem encerrar adequadamente pode deixar lento o COLETOR DE LIXO do firebird... então inicie e termine um processo... de preferência, encerre a conexão - isto vai trazer um benefício muito grande para o banco de dados..... AFINAL devemos escrever para o banco de dados, só assim ele responderá com eficiência.

Para fechar, "backup " você vai precisar um dia... faça testes para saber se ele é confiável - não deixe para descobrir quando for precisar. Não deixe-o no mesmo disco (de preferência, leve para casa);

Firebird 3.0 - Criando funções

#firebird

Estrutura

```
SET TERM ^ ;
create or alter function StartOfMonth(PDate date) returns date...
as -- retorna a data do primeiro dia do mes
begin
    return PDate - extract(day from PDate)+1;
end^
```

```
SET TERM ; ^
```

Utilizando a função

```
select StartOfMonth( cast('01/10/2016' as date) ) from RDB$Database
```

Delphi BDE interface para FireDac.

BDE acabou, o que fazer com todos aqueles aplicativos que usa BDE..? Alguém dirá, facil - vamos fazer conversão.
Penso que não. Quero rodar meus 50 aplicativos com 20 anos de vida com o Delphi mais recente.
Pensando nisto passei a trabalhar em um projeto intitulado de "fireTables". Agora tudo rodando nos compiladores mais novos....
Ver o código no GIT: https://github.com/amarildolacerda/FiredacTables

Firebird - CTE x Criando dimensão tempo

Emprestado de BI, fazer agrupamentos que mostre dados em dimensão de tempo é uma ferramenta de grande ajuda.

São diversas as situações onde é aplicável dimensionamento de dados por tempo - melhor utilizar exemplos:

Ex: Deseja saber qual dia da semana as vendas são mais concentradas - Seg, Ter, Qua, Qui, Sex, Sab, Dom....

```
with CTE as
(
   select data,wdia,cdia from DIM_DATE('12/01/2015','12/31/2015')
)
elect d.wdia as dia, d.cdia as cdia, sum(valor) as Valor
from cte d join TAB_VENDAS a on (d.data=a.data)
group by d.wdia,d.cdia
```

Código para a procedure DIM_DATE:

```
SET TERM ^ ;
CREATE OR ALTER PROCEDURE DIM_DATE (
       p_data_de date,
       p_data_ate date)
      returns (
           data date,
```

```
            dia integer,
            wdia integer,
            cdia varchar(1),
            mes integer,
             ano integer,
             semestre integer,
             trimestre integer,
             nsemana integer,
             inimes date,
             fimmes date)
 as
declare variable dt date;
begin
    dt = :p_data_de;
    while (dt <= :p_data_ate ) do
   begin
        data = :dt;
        dia = extract(day from :dt);

        wdia = extract(weekday from :dt); -- cdia base
        cdia = substr('DSTQQSS',wdia+1,wdia+1);

        mes = extract(month from :dt);
        ano = extract(year from :dt);
        semestre = case when mes<=6 then 1 else 2 end;
        trimestre = trunc((mes-1) / 3)+1;

        inimes = :dt - dia +1;
        fimmes = dateadd(month, 1,:inimes) - 1;
        nsemana = extract(week from :data);

       suspend;
       dt = dt+1;
  end

end^

SET TERM ; ^
GRANT EXECUTE ON PROCEDURE DIM_DATE TO SYSDBA;
```

#firebird #delphi

Firebird - Pulou alguma nota fiscal ?

#firebird #delphi #sql

O departamento fiscal quer saber se tem alguma nota fiscal faltando no banco de dados.
Cla...ro que não vamos ficar lendo uma lista para ver se tem alguma que pulou número.
Vamos perguntar para o banco de dados.

A mecânica não é muito trivial, já que não tem uma instrução que descubra algo desconhecido, então vamos preparar o banco para que conheça o problema a ser resolvido.

1) criar uma procedure selecionável que monte uma sequência esperada de números possíveis:

```
CREATE OR ALTER PROCEDURE DIM_INTEGER (
    nmin integer,
    nmax integer)
    returns (
        numero integer)
    as
   begin
         numero = nMin;
         while (numero<=nMax) do
         begin
           suspend;
           numero = numero+1;
         end
end
```

2) agora de posse de uma lista com os números esperados, podemos perguntar para o banco qual nota esta faltando em um intervalo (ex: entre 1000 e 2000):

```
select a.numero
from dim_integer( 1000 , 2000 ) a
where not exists (select notafiscal from tab_NotaFiscal b where b.numeroNotaFiscal=a.numero)
```

3)resultado:
Uma relação de números que não existem na tabela de nota fiscal;