Operações com campos datas e horas

Saiba como manipular datas e horas no FB

por ***Carlos H. Cantu***

Os campos *datem*, *time* e *timestamp* sempre foram cercados de um certo mistério, principalmente para os novos usuários. A introdução do dialeto 3 no IB 6.0 e a conseqüente mudança de nomenclatura e de significado dos campos DATE contribuíram ainda mais com a confusão que os usuários podem fazer entre os tipos. Nesse artigo, vou apresentar os 3 tipos de dados existentes para manipular data, hora e data+hora e mostrar exemplos práticos de sua utilização.

Tipos de dados disponíveis

Nas versões anteriores ao IB 6.0, o único tipo de dado existente para manipular data e/ou hora era o tipo DATE que armazenava o conjunto data+hora. Não havia um tipo específico para armazenar somente datas ou somente horas. Com o lançamento do IB 6.0 (e consequentemente no Firebird 1) e a introdução do dialeto 3, os usuários puderam desfrutar de novos tipos de dados que acabaram com essa "limitação".

O principal motivo da introdução de dialetos nos bancos de dados foi justamente oferecer um meio de disponibilizar novos tipos de dados e funcionalidades sem quebrar a compatibilidade com os bancos de versões anteriores ao IB 6. Um banco gerado em uma versão anterior ao IB 6 é tratado sempre como DIALETO 1. No dialeto 1, os antigos campos DATE são interpretados como TIMESTAMP e continuam sendo o único tipo de dado disponível para manipulação de datas e horas. Nos bancos criados no dialeto 3 existem 3 tipos de dados para manipulação de datas e horas, conforme a tabela abaixo:

Tipo	Descrição	Intervalo
TIMESTAMP	Substituto do antigo tipo DATE, armazena um par composto por DATA+HORA	Ver intervalos abaixo
DATE	Armazena somente uma DATA (composta por ano, mês e dia)	1/Jan/0001 a 31/Dez/9999
TIME	Armazena somente uma HORA (composta por hora, minutos, segundos e frações de segundos	00:00 AM à 23:59.9999 PM

Dica: O dialeto 1 é mantido apenas para efeito de compatibilidade. Todos os novos bancos devem ser criados no dialeto 3 que oferece mais recursos e facilidades.

No resto deste artigo vamos trabalhar somente com os tipos do dialeto 3. Note que tanto o tipo DATE, TIME como o TIMESTAMP são armazenados internamente no banco de dados como uma *longword* (32bits).

Modified Julian Date (MJD)

O InterBase e o Firebird utilizam um formato interno para cálculo de datas chamado MJD que consiste em uma versão modificada do sistema Juliano (JD - Julian dates). No sistema Juliano, o calendário começa ao meio-dia de 1/Jan/4713 ac. já no MJD ele começa em 17/Nov/1858 ou seja, 2400000.5 dias após a data inicial do sistema Juliano. Esse número não foi escolhido por acaso. Se você está curioso para saber mais sobre o MJD e o JD consulte os links https://pdc.ro.nu/mjd.html .

Criando campos Date, Time e TimeStamp

A sintaxe para a criação de campos date, time, e timestamp em uma tabela está descrita a seguir:

```
CREATE TABLE TESTE (
      C_DATA       DATE DEFAULT CURRENT_DATE,
      C_HORA       TIME DEFAULT CURRENT_TIME,
      C_TIMESTAMP  TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Na tabela TESTE definimos 3 campos utilizando os 3 tipos de dados disponíveis para manipulação de data e hora. Observe que no parâmetro DEFAULT utilizei as expressões SQL CURRENT_DATE, CURRENT_TIME e CURRENT_TIMESTAMP para atribuir respectivamente a data, hora e data+hora atuais quando esses valores não forem indicados durante a inserção de um registro.

Recuperando data e hora do sistema

O IB/FB fornece expressões e literais SQL que disponibilizam as informações do relógio do sistema para serem utilizadas dentro do banco de dados. Na tabela abaixo temos uma lista dessas expressões e o tipo do valor retornado por elas.

Variável	**Função**
CURRENT_DATE	Retorna a data atual do sistema (somente no dialeto 3)
CURRENT_TIME	Retorna a hora atual do sistema (somente no dialeto 3)
CURRENT_TIMESTAMP	Retorna a data e a hora do sistema (somente no dialeto 3)
NOW	Retorna a data e a hora do sistema
TODAY	Retorna a data atual do sistema
TOMORROW	Retorna a data do dia de amanhã
YESTERDAY	Retorna a data do dia de ontem

A **figura 1** mostra o resultado de todas essas variáveis depois da execução do seguinte select:

```
select
  current_date as "CURRENT_DATE",
  current_time as "CURRENT_TIME",
  current_timestamp as "CURRENT_TIMESTAMP",
  cast ('now' as timestamp) as "NOW",
  cast ('today' as date) as "TODAY",
  cast ('tomorrow' as date) as "TOMORROW",
  cast ('yesterday' as date) as "YESTERDAY"
from rdb$database
```

Dica: Para utilizar as literais *now, today, tomorrow* e *yesterday* em selects em bancos no dialeto 3 deve-se utilizar a função CAST, caso contrário se obterá um erro.

Usando strings para referenciar datas e horas

Podemos utilizar *strings* referenciando informações do tipo *time, date* e *timestamp* em comandos e operações no Banco de Dados. Devemos, no entanto, tomar alguns cuidados na hora de escrever essas *strings* para passarmos os valores no formato correto.

Para tipos TIME, devemos passar os valores formatados com a máscara **hh:mm:ss.mmm** (hora,minutos,segundos e milésimos de segundos) . Para o tipo DATE temos a possibilidade de passar as datas em diferentes formatos sendo que dependendo do caractere separador o servidor interpreta de maneira diferente o formato que está sendo utilizado na data, por exemplo, se quisermos passar a data no formato dia/mês/ano, devemos utilizar o ponto (.) como separador e não a barra. Exemplos diferentes de se passar a mesma data (20 de janeiro de 2003):

’01/20/2003’

’01.20.2003’
’20-jan-2003’
’20-January-2003’
’2003-01-20’

O tipo timestamp deve ser formatado utilizando as máscaras comentadas anteriormente para os tipos *date* e *time*. Sendo assim, também temos várias maneiras de referenciar uma mesma data+hora, no exemplo abaixo utilizaremos a data 20 de janeiro de 2003 e a hora 18:00:

‘20.01.2003 18:00:00.000’
‘01/20/2003 18:00’
‘01-Jan-2003 18:00:00’
‘2003-01-20 18:00’

Aritmética com data, hora e timestamp

Um dos usos mais comuns dos tipos que identificam “tempo” é realizar operações entre eles para se obter intervalos/períodos. A seguir apresentarei as operações mais comuns entre os tipos e a unidade de tempo fornecida no resultado.

Operações com datas

Diferença entre 2 datas

Ao realizarmos uma operação de subtração entre dois campos (ou duas variáveis) do tipo DATE obteremos um resultado que equivale ao número de dias entre o período informado. Um exemplo:

```
select cast('1.1.2004' as date) - cast('1.1.2003' as date)
from rdb$database
```

trará como resultado o valor (inteiro) 365, ou seja, temos um espaço de 365 dias entre as duas datas informadas.

Nota: Não é possível realizar uma soma entre duas datas.

Somando e subtraindo dias à uma data

Quando somamos ou subtraímos um número de uma data, estamos somando ou subtraindo DIAS. Sendo assim, ao executar o comando

```
select cast('1.1.2003' as date) + 30, cast('1.1.2003' as date) - 30
from rdb$database
```

o resultado serão as datas 31/Jan/2003 e 02/Dez/2002.

Operações com HORA (TIME)

Diferenças entre 2 horas

Quando subtraímos campos ou variáveis do tipo *time*, temos como resultado o espaço de tempo em segundos (e frações de segundos) do período indicado. Exemplo:

```
select cast('18:00:00.000' as time) - cast('12:00:00.000' as time)
from rdb$database
```

O resultado dessa operação é 21.600,0000 segundos, ou seja, 360 minutos ou 6 horas.

Nota: Não é possível realizar uma soma entre duas horas.

Operações com TIMESTAMP

Somas e subtrações com timestamps

Quando somamos ou subtraímos valores aos tipos *timestamp* esses valores representam um espaço de dias (e suas frações), por exemplo, se adicionarmos 1.5 a um *timestamp* estamos adicionando 1 dia e 12 horas, do mesmo modo que somando 1.25 estaremos adicionando 1 dia e 6 horas. Podemos também subtrair um *timestamp* de outro obtendo assim o intervalo de tempo entre eles também expresso em dias (e suas frações). Para entender melhor veja o exemplo:

```
select
  cast('1.1.2003 00:00:00.0000' as timestamp) - 1.5,
  cast('1.1.2003 00:00:00.0000' as timestamp) + 1.5,
  cast('2.1.2003 12:00:00.0000' as timestamp) - cast('1.1.2003 00:00:00.0000' as
timestamp)
from rdb$database
```

Ao se executar o comando acima, o resultado obtido será "30/Dez/2002 12:00", "02/Jan/2003 12:00" e 1.5 respectivamente.

Nota: Não é possível realizar uma soma entre dois *timestamps*.

Operações entre *date* e *time*

A única operação que você pode executar entre um tipo *date* e um tipo *time* é a **concatenação**. Nesse caso o resultado será um tipo *timestamp* composto da data informada no tipo DATE e da hora informada no tipo TIME. Exemplo :

```
select
  cast('1.1.2003' as date) + cast('23:59:59.9999' as time)
from rdb$database
```

O resultado do comando anterior será "1/Jan/2003 23:59:59.9999".

Usando a função EXTRACT

A função *Extract* é muito útil quando precisamos desmembrar os componentes que formam uma data, hora ou um timestamp. Com ela podemos recuperar somente a hora de um tipo time, ou somente o dia de uma data, ou somente o ano, etc.

A sintaxe da função *extract* é

Extract (*informação* from *valor*)

Os possíveis valores para o parâmetro *informação* bem como o tipo de resultado estão listados na tabela abaixo.

Informação	**Resultado**	**Faixa**
YEAR	SMALLINT	Ano, de 1 à 9.999
MONTH	SMALLINT	Mês, de 1à 12
DAY	SMALLINT	Dia, de 1 a 31
HOUR	SMALLINT	Horas, de 1 à 23
MINUTE	SMALLINT	Minutos, de 0 a 59
SECOND	DECIMAL(6,4)	Segundos, de 0 a 59.9999
WEEKDAY	SMALLINT	Dia da semana, de 0 a 6 onde 0 = domingo, 1 = segunda, etc.

YEARDAY	SMALLINT	Dia do ano, de 1 a 366

Uso prático para a aritmética das datas

Um dos usos mais comuns dos cálculos entre datas é obter informações sobre um período, como por exemplo, quantos dias, horas, minutos e segundos se passaram entre dois momentos. Para exemplificar os conceitos apresentados nesse artigo vamos criar agora uma *stored procedure* que recebe dois *timestamps* como parâmetros de entrada e retorna quantos dias, horas, minutos e segundos se passaram no intervalo determinado.

O código completo da procedure pode ser visto na **listagem 1**.

```
CREATE PROCEDURE INFO_PERIODO (
    INICIO TIMESTAMP,
    FIM TIMESTAMP)
RETURNS (
    DIAS INTEGER,
    HORAS INTEGER,
    MINUTOS INTEGER,
    SEGUNDOS INTEGER)
AS
DECLARE VARIABLE TEMP NUMERIC(18,7);
BEGIN
  /* Inicializa variáveis */
  dias = 0; horas = 0; minutos = 0; segundos = 0;
  /* Temp armazena a diferenca do periodo, em dias e frações de dias */
  temp = (fim - inicio);
  /* Pega o número de dias - parte inteira */
  if (temp > 0) then dias = temp - 0.5; else dias = temp;
  /* extraí os dias do timestamp, deixando apenas hora/min/seg' */
  temp = temp - dias;
  /* Um dia tem 86.400 segundos, assim transformamos o valor
     que temos em total em segundos */
  temp = temp * 86400;
  horas = (temp / 3600); /* Uma hora tem 3.600 segundos */
  if (horas > 0) then horas = horas - 0.5;
  temp = temp - (horas*3600); /* Temp vale agora minutos e segundos */
  minutos = (temp / 60); /* Um minuto tem 60 segundos */
  if (minutos > 0) then minutos = minutos - 0.5;
  segundos = temp - (minutos * 60); /* Para finalizar extrai os segundos */
END
```

Listagem 1. Procedure para retornar informações sobre um intervalo de tempo.

Inicialmente os parâmetros de retorno são inicializados com 0.

```
  dias = 0; horas = 0; minutos = 0; segundos = 0;
```

Em seguida o intervalo entre as datas é obtido e armazenado em uma variável intermediária chamada *temp*. Como visto anteriormente, uma operação de subtração entre 2 tipos *timestamp* retorna o resultado em dias (e sua frações). Para obtermos o número de dias desse intervalo devemos separar a parte inteira da variável *temp*, que representa o número de dias, da sua parte fracionária. Subtraímos 0.5 do valor de *temp* simulando um efeito de truncagem ao se converter o valor de *temp* para um inteiro (lembre-se que *temp* é uma variável do tipo *numeric* enquanto os parâmetros de retorno são todos inteiros). Depois fazemos com que *temp* fique somente com a parte fracionária do seu valor que representa o valor significando hora+minutos+segundos.

```
  temp = (fim - inicio);
  if (temp > 0) then dias = temp - 0.5; else dias = temp;
  temp = temp - dias;
```

Em seguida o que fazemos é uma série de cálculos para obter os valores horas, minutos e segundos restantes. Para facilitar os cálculos, transformamos o valor de *temp* em segundos, multiplicando-o por 86.400 que consiste no número total de segundos que temos em um dia (ou 24h). A partir daí fazemos uma série de divisões para obter os valores desejados. Veja que a cada divisão nós recalculamos o valor de *temp* de modo que ele contenha sempre o “resto” que nos interessa.

```
  temp = temp * 86400;
  horas = (temp / 3600); /* Uma hora tem 3.600 segundos */
  if (horas > 0) then horas = horas - 0.5;
  temp = temp - (horas*3600); /* Temp vale agora minutos e segundos */
  minutos = (temp / 60); /* Um minuto tem 60 segundos */
  if (minutos > 0) then minutos = minutos - 0.5;
  segundos = temp - (minutos * 60); /* Para finalizar extrai os segundos */
```

Quanto tempo você já viveu ?

Agora para completar o capítulo vamos criar um pequeno programinha em Delphi para chamar nossa *Stored Procedure* e exibir o resultado. A função do programa é mostrar quanto tempo você já viveu, desde a hora do seu nascimento até o momento atual (baseado no relógio do sistema).

Crie um aplicativo com somente um *form*, como mostrado na *figura 2*. Utilizaremos nesse exemplo o *dbExpress* como método de conexão com o banco de dados que contém nossa *stored procedure*. Para executar o exemplo precisamos apenas de um componente de conexão (TSQLConnection) e um outro para executar a *stored procedure* (TSQLStoredProc). O código principal do programa pode ser visto na *listagem 3*. Atente para o detalhe de que é necessário passar os valores para os parâmetros de entrada usando o formato SQLTimeStamp e não como um DateTime. Para isso utilizamos a função DateTimeToSQLTimeStamp para converter o formato DateTime para SQLTimeStamp.

A **figura 3** apresenta a tela do programa sendo executado.

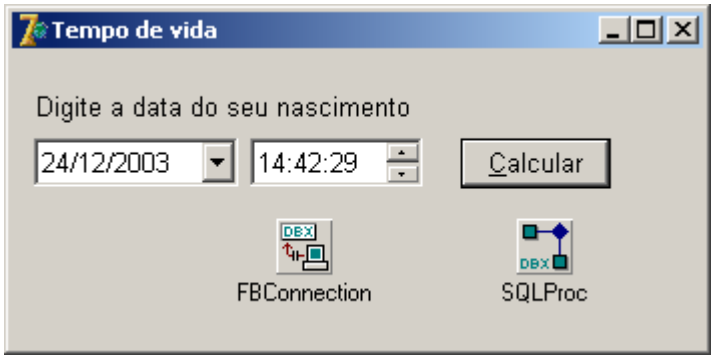
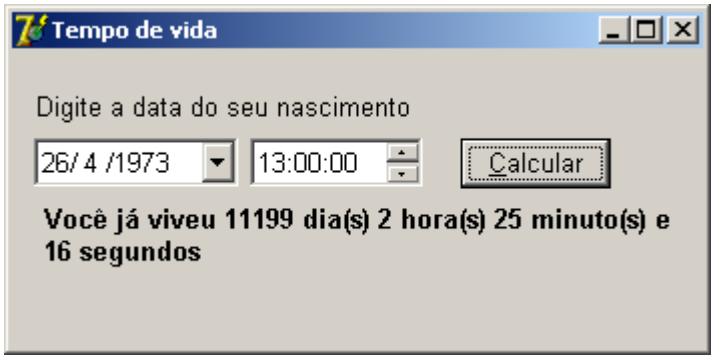


Figura2. *Form* contendo os controles necessários para a execução do programa.

```
procedure TForm1.Button1Click(Sender: TObject);
var st: string;
  Dias, horas, minutos, segundos: Integer;
  xData: TDateTime;
begin
  with SQLProc do
  begin
    xData := dtpData.DateTime;
    ReplaceTime(xData, dtpHora.Time);
    ParamByName('inicio').AsSQLTimeStamp := DateTimeToSQLTimeStamp(xData);
    ParamByName('fim').AsSQLTimeStamp := DateTimeToSQLTimeStamp(Now);
    ExecProc;
    Dias := ParamByName('dias').asInteger;
    Horas := ParamByName('horas').asInteger;
    Minutos := ParamByName('minutos').asInteger;
    Segundos := ParamByName('segundos').asInteger;
    st := 'Você já viveu ';
    if dias > 0 then st := st + IntToStr(Dias) + ' dia(s) ';
    if horas > 0 then st := st + IntToStr(Horas) + ' hora(s) ';
    if minutos > 0 then st := st + IntToStr(minutos) + ' minuto(s) ';
    st := st + 'e ' + IntToStr(segundos) + ' segundos';
    lbResult.caption := st;
  end;
end;
```

Listagem 3. Código principal do aplicativo exemplo.



Conclusão

Espero que com que esse artigo eu tenha conseguido desmistificar a manipulação de datas e horas no banco de dados. O segredo é entender o que cada tipo de dado representa e como ele armazena essas informações, bem como qual o resultado de cada tipo de operação aritmética realizada.

Esse artigo foi extraído do livro "Firebird Essencial", de minha autoria, e revisado em 20-dezembro-2021.
Autor: Carlos Henrique Cantu

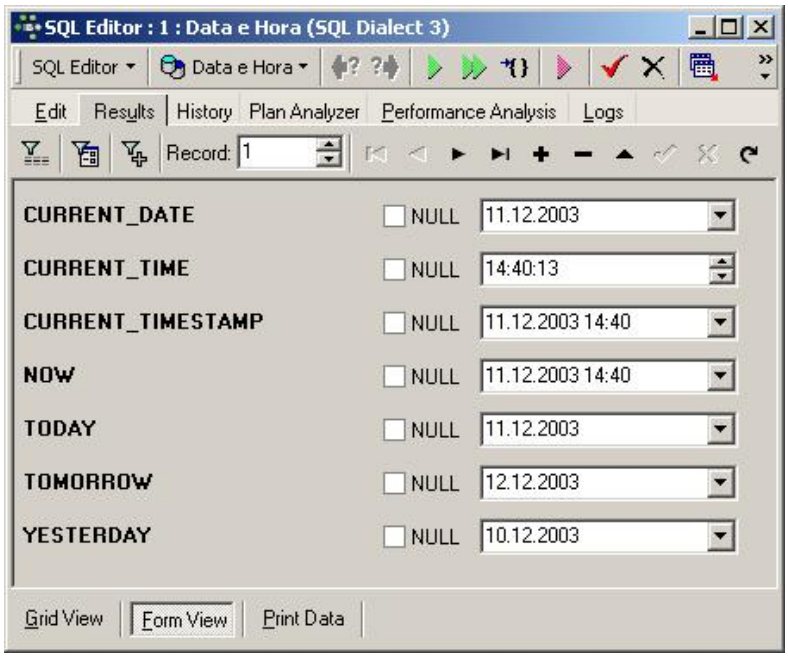


Figura1. Resultado das variáveis globais de data e hora.