Funções String em linguagem SP/Trigger

Todas estas funções podem ser implementadas facilmente por UDF, porém em algumas situações você não pode ou não quer usar UDF. Algumas razões para evitar o uso de UDFs são por exemplo:

- UDFs não são suportadas em NetWare
- UDFs são dependentes de plataforma (Sistema Operacional)
- Uma UDF perdida pode impedir que um banco de dados seja restaurado
- UDF não têm nenhuma informação sobre conjunto de caracteres em seus parâmetros
- UDFs não podem retornar <null>
- UDFs escritas incorretamente podem derrubar o servidor

TrimRight em SP

O InterBase não têm nenhuma função embutida para limpar espaços em branco. Tal função pode ser útil por exemplo para converter um campo CHAR para VARCHAR ao importar dados de arquivos externos. A função rtrim é por exemplo parte da biblioteca UDF padrão ib_udf.dll, mas às vezes você quer evitar usar UDFs. A menos que você precise usar isto em strings muito longas (por exemplo CHAR(30000)), é possível implementar a função trim como um procedimento armazenado (stored procedure).

Quando você LANÇAR uma string pequena única, a declaração terá sucesso se os caracteres removidos forem espaços, ou falhará (causará uma exceção) se você tentar remover caracteres que não são brancos. Tente somente lançar STRINGs pequenas e os erros pegos farão o truque. Ele pode ser chamado também como um procedimento armazenado (EXECUTE PROCEDURE TrimRight 'abc') ou como um procedimento de seleção (SELECT ... FROM TrimRight('abc')).

```
CREATE PROCEDURE TrimRight (str VARCHAR(10))
  RETURNS (ret VARCHAR(10)) AS
BEGIN
  ret = str;
  IF (str IS NULL) THEN BEGIN SUSPEND; EXIT; END
  IF (str = '') THEN BEGIN ret = ''; SUSPEND; EXIT; END
  BEGIN
    ret = CAST (str AS char(9));
    ret = CAST (str AS char(8));
    ret = CAST (str AS char(7));
    ret = CAST (str AS char(6));
    ret = CAST (str AS char(5));
    ret = CAST (str AS char(4));
    ret = CAST (str AS char(3));
    ret = CAST (str AS char(2));
    ret = CAST (str AS char(1));
    SUSPEND;
    WHEN ANY DO SUSPEND;
  END
END
```

Exemplos:

```
  SELECT '>' || ret || '<'
    FROM TrimRight (null)
  ============
  <null>


  SELECT '>' || ret || '<'
    FROM TrimRight (' 1234      ')
  ============
  > 1234<


  EXECUTE PROCEDURE TrimRight '1234      '
  ==========
  1234
```

Exemplo de uma chamada de outro procedimento armazenado (Stored Procedure):

```
  EXECUTE PROCEDURE TrimRight str_in
        RETURNING_VALUES    str_out;
```

Notas:

Não é possível usar o comando WHILE (...) DO para simplificar o código porque ele não pode conter variável no lugar do tamanho do caracter (Por exemplo: CAST(str AS CHAR(:len))).

Há um "bug" no IB 5.1/5.6 (corrigido no IB6 eno Firebird) - se você remover os parênteses do BEGIN/END interno, então SELECT ... FROM TrimRight(null); e SELECT ... FROM TrimRight(''); retornará duas linhas ao invés de uma.

Truncando uma string numa SP

Ao invés de limpar, truncar é uma função que encurtará a string apesar do seu conteúdo, por exemplo, ele irá remover até caracteres não-brancos. Não existe nenhuma função semelhante no Interbase (Exceto UDF externa). Quando lançar uma string longa como uma curta, o Interbase disparará uma exceção "... string truncation". Quando atribuir uma string longa diretamente a uma variável curta, o InterBase irá disparar uma exceção também, mas o valor truncado será atribuído de qualquer jeito! Tudo o que precisamos fazer é tratar a exceção pela instrução WHEN ANY DO. Aqui está um procedimento de exemplo que trunca uma string até 5 caracteres:

```
CREATE PROCEDURE Trunc10To5 (a varchar(10))
  RETURNS (ret varchar(5)) AS
BEGIN
  ret = '';
  ret = a;
  WHEN ANY DO EXIT;
END
```

Comando

```
  EXECUTE PROCEDURE Trunc10To5 '1234567890'
```

Retornará: '12345'.

Note que você não deve usar CAST, e aquela variável que você está atribuído não deve conter , assim estes dois procedimentos não trabalharão:

```
CREATE PROCEDURE test1 (a varchar(10))
  RETURNS (ret varchar(5)) AS
BEGIN
  ret = null;
  ret = a;
  WHEN ANY DO EXIT;
END

CREATE PROCEDURE test2 (a varchar(10))
  RETURNS (ret varchar(5)) AS
BEGIN
  ret = CAST(a AS VARCHAR(5));
  WHEN ANY DO EXIT;
END
```

Note também que isto é um "bug", pois o valor é atribuído mesmo se a exceção for disparada; porém esta é a forma como o IB4, IB5, IB6 e Firebird trabalham.

Função para obter o tamanho da string numa SP

Porque a função Length (tamanho) não modifica a string de estrada (como limpando ou truncando), a implementação usando o "loop" WHILE e o comando "LIKE" podem ser utilizados diretamente:

```
CREATE PROCEDURE Len (str VARCHAR(100))
  RETURNS (len INTEGER) AS
DECLARE VARIABLE pat VARCHAR(100);
BEGIN
  len = null;
  IF (str IS NULL) THEN EXIT;
  pat = '';
  len = 0;
  WHILE (NOT str LIKE pat) DO BEGIN
    pat = pat || '_';
    len = len + 1;
  END
END
```

Você pode omitir "len = null;" porque variáveis são inicializadas para "null" automaticamente. O tamanho será contado incluindo os espaços em branco.

```
  EXECUTE PROCEDURE Len null
     LEN
  =======
   <null>

  EXECUTE PROCEDURE Len ''
     LEN
  =======
        0

  EXECUTE PROCEDURE Len 'abc'
     LEN
  =======
        3
```

```
EXECUTE PROCEDURE Len 'xyz   '
     LEN
 =======
       6
```

Função para obter a posição de uma substring numa SP

Esta função devolve o índice do primeiro caracter numa substring especificada (Parâmetro SubStr) isso acontece em uma determinada string (Str).

```
CREATE PROCEDURE Pos (SubStr VARCHAR(100), Str VARCHAR(100))
  RETURNS (Pos INTEGER) AS
DECLARE VARIABLE SubStr2 VARCHAR(201); /* 1 + SubStr-lenght + Str-length */
DECLARE VARIABLE Tmp VARCHAR(100);
BEGIN
  IF (SubStr IS NULL OR Str IS NULL)
  THEN BEGIN Pos = NULL; EXIT; END
  SubStr2 = SubStr || '%';
  Tmp = '';
  Pos = 1;
  WHILE (Str NOT LIKE SubStr2 AND Str NOT LIKE Tmp) DO BEGIN
    SubStr2 = '_' || SubStr2;
    Tmp = Tmp || '_';
    Pos = Pos + 1;
  END
  IF (Str LIKE Tmp) THEN Pos = 0;
END
```

A variável Tmp é usada para parar o "loop" se o número de repetições for igual ao tamanho do STR. Porque o SubStr é usado no lado direito do operador LIKE, ele não deveria conter curingas SQL, por exemplo, '_' e '%' (a menos que você use a cláusula ESCAPE). Se substring não for encontrado, o valor de retorno é zero.

```
  EXECUTE PROCEDURE Pos 'ab', 'abcdefghij'
     POS
  =======
       1

  EXECUTE PROCEDURE Pos 'cd', 'abcdefghij'
     POS
  =======
       3

  EXECUTE PROCEDURE Pos 'x', 'abcdefghij'
     POS
  =======
       0
```

Função Substring numa SP

Eu deixarei isto como lição de casa para os estimados leitores.

Artigo Original: http://www.volny.cz/iprenosil/interbase/ Ivan Prenosil	
Tradução e adaptação: **Edison Jamir Benjamini** informatica@amauc.org.br	**Comunidade Firebird de Língua Portuguesa** **Visite a Comunidade em:** **http://www.comunidade-firebird.org**
A Comunidade Firebird de Língua Portuguesa foi autorizada pelo Autor do Original para elaborar esta tradução.	