Sumario:

Este artigo descreve a manipulação de dados LITERAIS, contendo os principais comandos de pesquisa, concatenação, trocas de textos. O abap é uma linguagem de fácil manipulação de dados, este artigos descreve os principais comandos usados no dia-dia de desenvolvimento ABAP/4.

Sobre o Autor:

Uderson Luis Fermino, formado em Ciências da Computação pela Faculdade de Pesquisa e Ensino IPEP, atua no mercado a 2 anos como desenvolvedor Java nas plataformas: (J2SE, J2EE e J2ME), com participação em grandes projetos envolvendo estas tecnologias. É consultor ABAP com experiências em REPORT, ALV (GRID, LIST, BLOCK, OO, TREE, HIERARQUICK), IDOC, ALE, ONLINE, SAPSCRIPT, SMARTFORM, NETWEAVER (JCO, BSP, WebDynpro).

Email:

Uderson@gmail.com

Manipulação de dados

O abap é uma fonte rica de ferramentas para trabalhar com manipulação de dados, existem diversa ferramentas, para estás manipulação, será descritos algumas suas funcionalidades e suas sintaxes:

- **Trocando palavras, textos ou frases:**

Para exemplificar, iremos fazer a troca de um (.) ponto por uma (,) vírgula, onde pode-se usar o comando TRANSLATE, ficando sua sintaxe:

TRASNLATE VARIAVEL **USING** '. ,', a troca é feita através da ordem....
Devido o ponto (.), estar antes da vrigula (,) o comando entende que é para fazer a troca do ponto pela vírgula.

O comando translate faz diversos tipos de trocas de textos dentro de um texto, este texto não necessita ser uma variável do tipo c ou string pode ser qualquer tipo de dado

Exemplo.:

Troca & por caractere SPAÇO.

```
DATA: TEXTOS TYPE STRING,
      TEXTOT TYPE STRING.

      TEXTOT = 'COMANDO&TRANSLATE&ABAP'.
      TEXTOS = TEXTOT.

      TRANSLATE TEXTOT USING '& '.

      WRITE: / 'PALAVRA ANTIGA: ', TEXTOS, ' TEXTO NOVO: ', TEXTOT.
```

Pode-se usar a seguinte estrutura:

ABBAabba onde for encontrado:

A será trocado para B
B será trocado para A
a será trocado para b
b será trocado para a.

Exemplo.:

B a r b c b d b a r b
A b r a c a d a b r a

```
DATA TEXT TYPE STRING.
TEXT = `Barbcbdbarb`.
TRANSLATE TEXT USING 'ABBAabba'.

WRITE: / TEXT.
```

Alterando tudo para maiúsculo

TRANSLATE abap TO UPPER CASE.

Este comando altera todos os caracteres do texto para maiúsculo.

Exemplo.:

```
DATA: TEXTOMI TYPE STRING,
    TEXTOMA TYPE STRING.

    TEXTOMI = 'Comando&Ttranslate&Aabap'.
    TEXTOMA = TEXTOMI.

    TRANSLATE TEXTOMA TO UPPER CASE.

    WRITE: / 'PALAVRA ANTIGA: ', TEXTOMI, ' TEXTO NOVO: ', TEXTOMA.
```

Alterando tudo para minusculo

TRANSLATE abap TO LOWER CASE.

Este comando altera todos os caracteres do texto para maiúsculos

```
DATA: TEXTOMIN TYPE STRING,
    TEXTOMAI TYPE STRING.

    TEXTOMAI = 'COMANDO&TRANSLATE&ABAP'.
    TEXTOMIN = TEXTOMAI.

    TRANSLATE TEXTOMIN TO LOWER CASE.

    WRITE: / 'PALAVRA ANTIGA: ', TEXTOMAI, ' TEXTO NOVO: ',
TEXTOMIN.
```

REPLACE

O comando replace é um trocador de caractere, a finalidade deste comando é consumir caractere e realizar trocas, vamos ver suas derivações.

REPLACE ‘texto a ser alterado no texto’ **IN** variável **WITH** ‘texto que irá substituir’.

Trocando o ponto pela vírgula, observe que está sintaxe somente troca a primeira ocorrência, neste exemplo se tivéssemos mais de uma vírgula somente a primeira vírgula seria trocada.

```
DATA: V_VALOR TYPE STRING VALUE '50.0000'.

REPLACE '.' IN V_VALOR WITH ','.

WRITE: / V_VALOR.
```

Trocar um pedaço de texto (substring).

Neste exemplo iremos trocar a palavra ABAP por Advanced Business Application Programming.

```
DATA: V_TEXTO TYPE STRING VALUE 'REPLACE COM ABAP DA SAP'.

REPLACE 'ABAP' IN V_TEXTO WITH 'Advanced Business Application Programming'.

WRITE: / V_TEXTO.
```

Este comando apenas troca uma ocorrência, para que todas as ocorrências encontradas no texto sejam trocadas é necessário utilizar a seguinte sintaxe do replace:

REPLACE ALL OCCORRENCES OF ‘texto a ser alterado no texto’ **IN** variável **WITH** ‘texto que irá substituir’.

Exemplo:

```
DATA: V_TODAS TYPE STRING VALUE 'O CBAP É CMA LINGUCGEM MUITO BOC'.

REPLACE ALL OCCURRENCES OF 'C' IN V_TODAS WITH 'A'.

WRITE: / V_TODAS.
```

Pode acontecer que exista uma palavra ou caractere que seja necessário fazer a troca, porém pode acontecer que a palavra esteja com caracteres diferenciados, em minúsculo e maiúsculo.

Exemplo.;

Trocar a palavra: ABAP por PABA porém caso no texto exista a palavra abap ou Abap ou aBap e assim por diante estas palavra serão trocadas também pois o comando não diferencia minúsculo e maiúsculo, para que somente a palavra ABAP seja trocada é necessário informar o parâmetro **RESPECTING CASE,** ficando.:

REPLACE ALL OCCORRENCES OF 'texto a ser alterado no texto' **IN** variável **WITH** 'texto que irá substituir' **RESPECTING CASE.**

Desta forma somente será trocado a palavra contendo os respectivos caractere iguais.

Exemplo.:

```
DATA: V_TODASC TYPE STRING VALUE 'O CBAP É cMA LINGUCGEM
MUITO BOC'.

REPLACE ALL OCCURRENCES OF 'C' IN V_TODASC WITH 'A'
RESPECTING CASE.

WRITE: / V_TODASC.
```

Parâmetros adicionais

REPLACEMENT COUNT {variavel do tipo i}

Grava em uma variável do tipo i a quantidade de ocorrências alteradas.

REPLACEMENT OFFSET

Grava em uma variável do tipo i a ultima ou única ocorrência trocada, em caso do um caractere armazenará a posição do caractere e em caso de uma palavra ou frase armazenará a posição do primeiro caractere.

REPLACEMENT LENGTH.

Grava em uma variável do tipo i o tamanho da palavra trocada.

Exemplo.:

```
DATA: V_TEXTOR TYPE STRING VALUE 'O ABAP É UMA LINGUAGEM
MUITO BOA TESTE A'.

REPLACE ALL OCCURRENCES OF 'UMA' IN V_TEXTOR WITH 'uma'
          RESPECTING CASE
          REPLACEMENT COUNT  cnt
          REPLACEMENT OFFSET off
          REPLACEMENT LENGTH len.


WRITE: / V_TEXTOR, ' ', CNT, ' ', OFF, ' ', LEN.
```

SHIFT

Este comando serve para realizar retirada de caracteres de um texto, com este comando pode-se retira caractere da esquerda ou da direita, de uma substring.

Sintaxe:

Retirar um determinado caractere da esquerda do texto:

SHITF {texto} LEFT DELEATING LEADIND {character a ser retirado}.

Retirar um determinado caractere da direita do texto

SHITF {texto} RIGHT DELETING TRAILING {character a ser retirado}.

Exemplo.:

```
DATA: VARIAVEL  TYPE STRING VALUE '0000202030000',
        VARI_ZERO TYPE C VALUE '0'.

SHIFT VARIAVEL LEFT  DELETING LEADING  VARI_ZERO.
SHIFT VARIAVEL RIGHT DELETING TRAILING VARI_ZERO.

WRITE: / VARIAVEL.
```

Onde a variável “vari_zero”, pode ser trocada para qualquer caractere, inclusive o espaço, lembrando que o caractere espaço pode ser trocado pelo comando SPACE.

O comando SHIFT permite retirar os zeros a direita, onde é somente trocar a posição LEFT por RIGHT, com este comando é possível retirar qualquer dado das extremidades.

Pode-se especificar mais de um caractere para ser retirado.

Exemplo:

Retirar todos os caracteres “T E S T E ” da direita.

```
DATA TTtext TYPE string VALUE 'ABAP É UMA LINGUAGEM FACIL TESTE'.

SHIFT TTtext RIGHT DELETING TRAILING 'T E S T E'.

WRITE / TTtext.
```

- **BY ... PLACES**

  Com este argumento do comando SHIFT é retirado todos os caractere de 1 até o especificado.

  Sintaxe.:

  SHIFT { texto } BY { posição } PLACES

  Será retirado todos os caracteres do “TEXTO” especificado da posição 1 até a posição N, onde a posição N pode variar de 1 até o tamanho maximo da literal.

  Exemplo.:

  Será retirado todos os caracteres da literal “SAP TESTE SHIFT BY PLACE” da posição 1 até a posição 10.

```
DATA: TEXTPLACE TYPE STRING VALUE 'ABAP SAP TESTE SHIFT BY
PLACE',
    POS  TYPE I.
```

```
FIND 'SHIFT' IN TEXTPLACE MATCH OFFSET POS.

SHIFT TEXTPLACE BY POS PLACES.

    WRITE / TEXTPLACE.
```

- **UP TO**

Este comando retira todos os caracteres da posição 0 até o encontrar aquela substring.

Exemplo.:

```
DATA: TEXTUPTO TYPE STRING VALUE 'ABAP SAP TESTE SHIFT BY
PLACE'.

SHIFT TEXTUPTO UP TO 'SHIFT'.

    WRITE / TEXTUPTO.
```

O comando SHIFT, permite fazer a troca de texto em forma circular,

```
DATA: ALPHA1(11) TYPE C VALUE 'ABCDEFGHIJ',
     ALPHA2     TYPE STRING.
ALPHA2 = ALPHA1.

   DO.

    IF SY-INDEX = 12.
      EXIT.
    ENDIF.
          SHIFT ALPHA1 CIRCULAR.
          SHIFT ALPHA2 CIRCULAR.
   ENDDO.
```

Pack

O comando Pack serve para retira os 0 (zeros) a esquerda de uma variável, porem existe um problema, ele troca os 0s (zeros ou zero), por espaços, isso faz com que uma variável, por exemplo: data: v_esp_zero(10) type c value '0000095678'. Caso o comando seja usado nesta variável, ela ficará assim " 95768", para resolver este problema existe o comando CONDENSE ou o comando SHIFT que retira todos os espaços de uma variável.

Sintaxe do Pack

PACK variavel_com_zeros TO variavel_sem_zero.

Sintaxe do condense:

CONDENSE variavel_retira_espaço.

Exe.:
data: v_esp_zero(10) type c value '0000095678'
v_esp_sem_sero(10) type c.

pack v_esp_zero to v_esp_sem_sero. >> ficando ' 95678'
condense v_esp_sem_sero. >> ficando '95678'.

SEARCH

Pesquisa em Strings (Textos), o comando SEARCH , procura determinadas ocorrências dentro de um textos.

Existe diversa sintaxe, vejamos as mais utilizadas:

SEARCH TEXTO FOR ‘PALAVRA/SILABA/FRASE’ AND MARK.

Esta sintaxe retorna a variável de sistema sy-subrc igual a 0 (zero), caso o texto foi encontrado, e no texto transfere para maiúsculo a parte procurada estiver em minúsculo ou transfere para minúsculo caso a parte procurada estiver em maiúsculo.

Exemplo:
Data: v_texto type string value ‘Uderson Luis é o autor deste tutorial’.

```
SEARCH v_texto for 'fato' AND MARK.
    if sy-subrc = 0.
      write: 'Encontrou', v_texto
    else.
      write: 'não encontrou'.
    endif.
```

Como a palavra procurada existe no texto será impresso a FRASE:

Encontrou **UDERSON** Luis é o autor deste tutorial

SEARCH TEXTO FOR ‘palavra-abraviada’ ABBREVIATED.

Esta sintaxe retorna a variável de sistema sy-subrc igual a 0 (zero), caso o texto foi encontrado, ela verificará se existe uma palavra que tenha está abreviação

Data: frase(50) type c value ‘Alaska Texas e California’.

```
SEARCH FRASE FOR ‘CLFRN’ ABBREVIATED.
    if sy-subrc = 0.
       write: 'Encontrou', v_texto
    else.
       write: 'não encontrou'.
    endif.
```

FIND

Pesquisa em Strings (Textos), o comando FIND , procura determinadas ocorrências dentro de um textos.

Existe diversa sintaxe, vejamos as mais utilizadas:

FIND

{TEXTO A SER PESQUISADO}

IN

{VARIAVEL DO TIPO STRING OU CHAR}

MATCH OFFSET

{VARIAVEL DO TIPO I }.

Está sintaxe pesquisa em texto literal um determinado caractere, substring (palavra, frase) e retorna a posição atual do literal pesquisado, se o literal for um caractere será retornada a posição atual do caractere, caso seja uma substring (palavra ou frase), será retornado a posição do primeiro caractere da substring.

Exemplo.:

```
DATA: ON TYPE I.
   FIND 'STRING' IN TEXTO MATCH OFFSET ON.
   WRITE / ON.
```

Pode acontecer que exista uma palavra ou caractere que seja necessário fazer a pesquisa, porém pode acontecer que a palavra esteja com caracteres diferenciados, em minúsculo e maiúsculo.

Exemplo.;

Trocar a palavra: ABAP por PABA porém caso no texto exista a palavra abap ou Abap ou aBap e assim por diante estas palavra serão trocadas também pois o comando não diferencia minúsculo e maiúsculo, para que somente a palavra ABAP seja trocada é necessário informar o parâmetro **RESPECTING CASE,** ficando.:

FIND 'texto a ser alterado no texto' **IN** variável **RESPECTING CASE.**

Desta forma somente será trocado a palavra contendo os respectivos caracteres iguais.

Exemplo.:

```
DATA: ON TYPE I.

 FIND 'MUITO' IN 'ABAP É MUITO FACIL' RESPECTING CASE MATCH
OFFSET ON.

   WRITE: / 'POSIÇÃO:  ', ON.
```

Parâmetros adicionais

MATCH COUNT {variável do tipo i}

Grava em uma variável do tipo i a quantidade de ocorrências alteradas.

MATCH OFFSET {variável do tipo i}

Grava em uma variável do tipo i a ultima ou única ocorrência trocada, em caso do um caractere armazenará a posição do caractere e em caso de uma palavra ou frase armazenará a posição do primeiro caractere.

MATCH LENGTH {variável do tipo i}.

Grava em uma variável do tipo i o tamanho da palavra trocada.

CONCATENATE

Este comando como o próprio nome diz serve para concatenar (juntar os dados) literais, por exemplo as literais “ABAP” “UMA” “ LINGUAGEM” “FACIL”.

Sintaxe.:

```
CONCATENATE
        TEXTO0 ... TEXTON
        INTO
 {VARIAVEL_QUE_RECEBERÁ_TEXTO_CONCATENADO}
```

Exemplo.:

```
DATA: TEXTO01 TYPE STRING VALUE 'ABAP',
     TEXTO02 TYPE STRING VALUE 'UMA',
     TEXTO03 TYPE STRING VALUE 'LINGUAGEM',
     TEXTO04 TYPE STRING.


CONCATENATE TEXTO01 TEXTO02 TEXTO03 INTO TEXTO04.

    WRITE / TEXTO04.
```

A impressão será ABAPUMALINGUAGEM.

Observe que as concatenações das literais ficaram todas juntas, para que as literais estejam separadas é necessário informar o argumento: SEPARATED BY { caractere } . Onde o caractere pode ser qualquer caractere este caractere é que definirá a separação.

Exemplo.:

```
CONCATENATE TEXTO01 TEXTO02 TEXTO03 INTO TEXTO04
SEPARATED BY SPACE.

    WRITE / TEXTO04.

CONCATENATE TEXTO01 TEXTO02 TEXTO03 INTO TEXTO04
SEPARATED BY '*'.

    WRITE / TEXTO04.
```

A impressão será ABAP*UMA*LINGUAGEM.

Pose-se usar o caractere SPACE ficando.:

```
CONCATENATE TEXTO01 TEXTO02 TEXTO03 INTO TEXTO04
SEPARATED BY SPACE.

   WRITE / TEXTO04.

CONCATENATE TEXTO01 TEXTO02 TEXTO03 INTO TEXTO04
SEPARATED BY SPACE.

   WRITE / TEXTO04.
```

A impressão será ABAP UMA LINGUAGEM.

Observação.:

O comando CONCATENATE, concatena os dados de acordo as variáveis, ou literais, caso a literal esteja desta forma.:

‘ ABAP’ ‘ é’ ‘ Uma’ ‘ LINGUAGEM’

Observe que cada literal contém espaços antes, ao concatenar estas literais a impressão será:

ABAP é Uma LINGUAGEM

Exemplo:

```
CONCATENATE ' ABAP' ' é' ' Uma' ' LINGUAGEM' INTO
TEXTO04.

   WRITE / TEXTO04.
```

Porém caso os espaços estejam depois dos caracteres alfa o comando CONCATENATE não consegue guardar os espaços.

‘ABAP ’ ‘é ’ ‘Uma ’ ‘LINGUAGEM ’

A saída será:

ABAPéUmaLINGUAGEM

Exemplo:

```
CONCATENATE 'ABAP    ' 'é   '    'Uma  ' 'LINGUAGEM ' INTO
TEXTO04.

   WRITE / TEXTO04.
```

Para resolver este problema na versão 6.0 existe um novo argumento que é:

RESPECTING BLANKS

Onde este argumento conserva os espaços em braços indiferentes se estejam antes dos caracteres alfas ou depois.

Exemplo 1.:

```
DATA: TEXTOA TYPE C LENGTH 20 VALUE 'UDERSON',
      TEXTOB TYPE C LENGTH 20 VALUE 'LUIZ',
      TEXTOC TYPE C LENGTH 20 VALUE 'FERMINO',
      TEXTOD TYPE C LENGTH 80.

CONCATENATE TEXTOA TEXTOB TEXTOC INTO TEXTOD RESPECTING
BLANKS.

   WRITE / TEXTOD.
```

Observe que as variáveis têm um tamanho de 20 caracteres, porém cada variável esta uma literal que não atige o tamanho maximo.

TEXTOB TYPE C LENGTH 20 VALUE 'LUIZ'

LUIS somente tem 4 caractere ficando a variável ‘LUIS ’, e assim por diante com as demais variáveis. Ao concatenar estas variáveis como os espaços em brancos são depois dos caracteres alfas, os comandos concatenate irá juntar os caracteres, para que seja conservado o tamanho da variável e os espaços que estiverem em brancos é necessário utilizar o argumento **RESPECTING BLANKS** juntamente do comando CONCATENATE.

Lembrando que para variáveis do tipos String não é valido, pois uma string, não possui tamanho fixo conforme uma variável do tipo C que é definido os tamanho antes de usar.

Ou pode-se usar uma literal de forma ‘ LITERAL ’ onde a literal estará dentro de ‘ aspa simples ’.

STRING e SUBSTRING

Antes de falar sobre a STRING é necessário que o leitor saiba que o cada literal é um vetor de caractere, exemplo:

U	D	E	R	S	O	N		L	U	I	S			

Está literal pode-se criar:

Data NOMEC type C length 15 value ‘UDERSON LUIS’.

Para literais do tipo STRING é a mesma coisa, porém o tipo STRING é um literal sem tamanho do vetor, podem-se manipular dados de tamanhos alternados.

SUBSTRING

SUBSTRING, é um pedaço de uma literal para coletar pedaço ou fatias de uma literal em abap é necessário informar os índices:

Pedaço inicial:

LITERAL({posição_final})

NOMEC(7) pega os caracteres de 1 até 7, informando o numero será coletados os caracteres iniciais até uma determinada posição.

Exemplo:

```
DATA NOMEC TYPE C LENGTH 15 VALUE 'UDERSON LUIS'.

WRITE:  NOMEC(7).
```

Pedaço Dinâmico:

LITERAL+{posição_inicial}({posição_final})

Este comando coleta uma substring onde é informada a posição inicial e a posição final.

Exemplo.:

```
DATA NOMEC TYPE C LENGTH 15 VALUE 'UDERSON LUIS'.

WRITE:  NOMEC+2(7).
```

Pedaço Final

LITERAL+{posição_inicial}

Esta comando coleta um pedaço da literal onde é informado a posição inicial , e será coletado da posição inicial passada até o final da literal.

Observação:

Deve-se tomar cuidado para não ultrapassar o tamanho da literal.

STRLEN

Este comando retorna o tamanho de uma string ou de um vetor de caractere, no caso de vetor de caractere não importa o tamanho fixo que foi definido, este comando retornará o tamanho de caractere existente no vetor.

Exemplo:

```
DATA: TEXTOTAM TYPE STRING VALUE 'ABAP É UMA LINGUAGEM
FACIL',
    TAMANHO  TYPE I.

    TAMANHO  = STRLEN( TEXTOTAM ).

    WRITE: TAMANHO.
```