Utilitários de JCL

Índice

DFSORT

Este manual foi elaborado com o intuito de fornecer algumas informações sobre DFSORT, portanto o mesmo não deve ser utilizado como referência e consequentemente não contém todas as informações que há nos manuais fornecidos pela IBM.

Qualquer dúvida ou a falta de detalhes sobre o assunto, favor consultar os manuais, visto que este material é destinado a um curso de inicialização ao DFSORT.

Capítulo 1

O QUE É DFSORT ?

DFSORT (Data Facility Sort) é um programa desenvolvido pela IBM para organizar informações. Pode-se usá-lo para classificar, fundir ou copiar um ou mais arquivos de entrada, obtendo-se um ou mais arquivos de saída.

CLASSIFICAÇÃO DE ARQUIVOS

A função primária do **DFSORT** é classificar registros de arquivos. A classificação pode ser ascendente ou descendente.

```
                (  ASCENDENTE  {8 6 2 7 1 4} ---> DFSORT ----> {1 2 4 6 7 8}
CLASSIFICAÇÃO:  (
                (  DESCENDENTE {8 6 2 7 1 4} ---> DFSORT ----> {8 7 6 4 2 1}
```

Quando o **DFSORT** está classificando pode-se:

- incluir ou excluir alguns registros do arquivo de entrada
- reformatar os registros
- somar valores em registros
- alterar a ordem natural de sequência

FUSÃO DE ARQUIVOS

Outra função do **DFSORT** é *merge* (fusão) de arquivos. Esta função consiste na fusão de dois ou mais arquivos, cujos registros estão **classificados** da mesma forma, para formar um ou mais arquivos, também classificados.

```
        {1 4 6 7 8}
FUSÃO:  (           ---> DFSORT ----> {0 1 2 3 4 5 6 7 8 9}
        {0 2 3 5 9}
```

Quando o **DFSORT** está executando o *merge*, pode-se:

- incluir ou excluir alguns registros do arquivo de entrada.
- reformatar os registros.
- somar valores em registros.
- alterar a ordem natural de sequência.

Se os arquivos de entrada não estiverem classificados, o *sort* pode classificá-los e fundí-los, de uma só vez, através da concatenação dos arquivos no SORTIN; não esquecer que arquivos com BLKSIZEs diferentes, devem ser concatenados na ordem decrescente de seus BLKSIZEs.

CÓPIA DE ARQUIVOS

A terceira grande função do **DFSORT** é copiar registros de arquivos sem qualquer classificação ou fusão. Pode-se copiar arquivos da mesma forma que se faria para classificá-los ou fundí-los.

CÓPIA: **{1 2 3 4 5}** ---> **DFSORT** ----> **{1 2 3 4 5}**

Quando o **DFSORT** está copiando pode-se:

- incluir ou excluir alguns registros dos arquivos de entrada
- reformatar os registros.

Pode-se fazer tudo o que foi dito acima, com apenas alguns cartões de controle do **DFSORT**. As funções básicas de classificação, fusão ou cópia são simples de serem solicitadas.

Pode-se executar o **DFSORT** via JCL ou via programa em Cobol, PL1 ou Assembler.

Capítulo 2

USANDO O DFSORT

Uma das formas de utilizar o **DFSORT** é via JCL, isto é, *sort* externo.

Para efeito de utilização do **DFSORT**, teremos um arquivo de teste chamado *BOOKSTORE* que contém alguns registros de tamanho fixo, e cada registro contém 12 campos (título do livro, último nome do autor, etc...).

Veremos neste capítulo que tudo o que se faz com o *sort* pode ser feito com o **DFSORT**, sem alterações nos cartões de controle, isto é, o *sort* é sub-conjunto do **DFSORT**.

CARTÃO DE CONTROLE SORT

Quando se necessita classificar um arquivo é natural descrever seus campos de classificação e a ordem em que eles devem ser classificados, para isto utiliza-se o cartão de controle chamado SORT.

Suponhamos que se queira classificar o nosso arquivo de teste pelo campo departamento, para isso:

1. deixar pelo menos um branco, e escreva SORT.
2. deixar pelo menos um branco, e escreva FIELDS=.
3. escrever, entre parenteses, e separando por vírgulas: aonde o campo departamento começa, relativo ao começo do registro (a 1ª. posição é byte 1; o campo departamento começa no byte 110); o tamanho do campo departamento em bytes (5 bytes); um código para o formato do dado (o campo departamento contém dados em caracteres, que se codifica como **CH**; a letra **A**, que significa classificação em ordem ascendente.

Veja a codificação abaixo:

```
      1 2                                                                  71  80
        SORT FIELDS=(110,5,CH,A)
                     --- - -- -
                      |   |   | |
                      |   |   | --------------------> ORDEM ASCENDENTE
                      |   |   ----------------------> DADOS EM CARACTERES
                      |   ------------------------->  TAMANHO DO CAMPO DEPART.
                      ---------------------------->  COMEÇO DO CAMPO DEPART.
```

Obs: a codificação do cartão tem que ser entre as colunas 2 e 71

Os formatos de dados mais comuns e seus códigos são:

FORMATO	CÓDIGO
EBCDIC	CH
ISCII/ASCII	AC
BINÁRIO	BI
DECIMAL ZONADO	ZD
DECIMAL COMPACTADO	PD

CLASSIFICAR REGISTROS POR VÁRIOS CAMPOS

Dentro de cada departamento pode-se classificar os registros pela especificação de mais campos. Quando se especificam dois ou mais campos de classificação, codifica-se do campo da ordem de maior prioridade para a menor.

Se forem especificados os campos na ordem abaixo:

- departamento do curso
- número do curso
- último nome do instrutor
- primeiro nome do instrutor
- título do livro

o cartão SORT pode ser codificado da seguinte forma:

```
        SORT FIELDS=(110,5,CH,A,115,5,CH,A,145,15,CH,A,160,2,CH,A,1,75,CH,A)
                     -----      -----      ------      -----      ----
                       |          |          |           |         |
DEPART. DO CURSO <------           |          |           |         |
NÚMERO DO CURSO  <-----------------           |           |         |
PRIMEIRO NOME DO INSTRUTOR <------------------            |         |
ÚLTIMO NOME DO INSTRUTOR <--------------------------------          |
TÍTULO DO LIVRO <-------------------------------------------------
```

Pode-se obter o mesmo resultado, com maior eficiência, se se especificar departamento e número do curso como um campo, e último nome do instrutor e primeiro como outro campo.

Se todos os campos tem o mesmo tipo de formato, pode-se específicar o formato dos dados utilizando o parâmetro FORMAT=.

Portanto, a codificação do cartão SORT seria da seguinte forma:

```
        SORT FIELDS=(110,10,A,145,17,A,1,75,A),FORMAT=CH
                     ------   ------   ----
                       |         |       |
                       |         |       -----> TÍTULO DO LIVRO
                       |         -----> ÚLTIMO E PRIMEIRO NOME DO INSTRUTOR
                       ---------------> DEP. E NÚMERO DO CURSO
```

CONTINUAÇÃO DO CARTÃO DE CONTROLE

Se o comando SORT não couber num cartão (ou qualquer outro comando do **DFSORT**) entre as colunas 2 e 71, pode-se continuá-lo no próximo cartão.

Se se terminar uma linha com uma vírgula seguida de um branco, o **DFSORT** assumirá que a próxima linha é de continuação. A continuação pode começar em qualquer lugar entre as colunas 2 e 71

Ex:

```
    SORT FIELDS=(110,10,A,145,17,A,
                 1,75,A),FORMAT=CH
```

Se se terminar uma linha com uma vírgula na coluna 71 seguida de um carater não branco, a continuação deve ser na coluna 16.

COMENTÁRIOS

Para inserir um comentário, colocar um asterisco na coluna de uma linha.
Ex:

```
* CLASSIFICACAO DESCENDENTE POR DEP. E NUMERO DE CURSO
```

CLASSIFICAÇÃO EM ORDEM DESCENDENTE

Para classificar em ordem descendente, especifique **D** em vez de **A**. Pode-se classificar alguns campos em ordem ascendente e outros em ordem descendente.
Por exemplo, para classificar departamento em ordem crescente e preços em cada departamento em ordem descendente:

```
        SORT FIELDS=(110,5,CH,A,170,4,BI,D)
                     --------    -------
                        |           |
                        |           ---------> PREÇO
                        ---------------------> DEPARTAMENTO
```

JCL PARA EXECUTAR O DFSORT

O JCL necessário para executar o **DFSORT** depende de como ele será utilizado, isto é, *sort* interno ou *sort* externo.

SORT EXTERNO

```
//STEP    EXEC PGM=SORT
//SORTLIB  DD  DSN=CTB.SORTLIB,DISP=SHR
//SYSOUT   DD     arquivo de saída para mensagens do DFSORT
//SORTIN   DD     arquivo(s) de entrada para o DFSORT
//SORTWKnn DD     define as áreas de trabalho para o DFSORT; nn pode estar fora de
                  ordem (nn entre 00 e 99 com blockset, senão os 32 primeiros serão
                  utilizados)
//SORTOUT  DD     define o arquivo de saída do DFSORT
//SYSIN    DD     define os cartões de controle para o DFSORT
//SORTCNTL DD     idêntico ao SYSIN
```

Obs: 1. O SORTIN pode ser a concatenação de vários arquivos.
2. Podem existir cartões de comentários "dentro" do SYSIN ou SORTCNTL, compostos por linhas em branco das colunas 1 a 71 ou com * na coluna 1.
3. Pode se usar o cartão END "dentro" do SYSIN ou SORTCNTL, o qual funciona como *End-of-File* (útil durante testes).
4. O cartão SORTLIB aponta para a biblioteca que contém módulos especiais do **DFSORT**. Este cartão somente é necessário quando a área de trabalho para o *sort* é em fita ou quando o *merge* não usa a técnica *blockset*. Portanto, dependendo do caso, este cartão é opcional.

```
----------------------------------------
EXEMPLO DE JCL PARA EXECUTAR SORT EXTERNO
----------------------------------------

//EXAMP JOB A492,PROGRAMMER
//SORT    EXEC PGM=SORT,REGION=512K
//SYSOUT   DD  SYSOUT=*
//SORTIN   DD  DSN=BOOKS.INPUT,DISP=OLD
//SORTWK01 DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SORTOUT  DD  DSN=BOOKS.OUTPUT,DISP=(,CATLG,DELETE),
//         SPACE=(CYL,(1,1)),UNIT=SYSDA
//SYSIN    DD    *
       SORT FIELDS=(110,10,A,145,17,A,1,75,A),
       FORMAT=CH
/*


------------
SORT INTERNO
------------
```

Para executar *sort* interno o **DFSORT** deve ser chamado por um programa, que pode ser Cobol, PL1 ou Assembler.

```
----------------------------------------
EXEMPLO DE JCL PARA EXECUTAR SORT INTERNO
----------------------------------------
//EXAMP   JOB A492,PROGRAMMER
//BOOKS   EXEC PGM=COBOLPGM,REGION=521K
//STEPLIB  DD  DSN=USER.PGMLIB,DISP=SHR
//SYSOUT   DD  SYSOUT=*
//MASTIN   DD  DSN=BOOKS.INPUT,DISP=OLD
//SORTWK01 DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//MASTOUT  DD  DSN=BOOKS.OUTPUT,DISP=(,CATLG,DELETE),
//             SPACE=(CYL,(1,1)),UNIT=SYSDA
//PRINTFL  DD  SYSOUT=*
/*
```

Capítulo 3

FILTRAR REGISTROS DO ARQUIVO DE ENTRADA

Freqüentemente, apenas um sub-conjunto de registros do arquivo de entrada (SORTIN) é necessário para a aplicação. Veremos como selecionar tais registros usando **DFSORT**.

É óbvio que esta facilidade aumenta a velocidade de classificação, porque os registros desnecessários não serão levados em conta para a ordenação.

ELIMINANDO OU SELECIONANDO REGISTROS INICIAIS

Se se quizer eliminar alguns registros iniciais do seu arquivo de entrada, pode-se acrescentar ao cartão de contole SORT a opção 'SKIPREC' (*Skip Records*)
Ex: SORT FILEDS=(...),SKIPREC=20
Neste exemplo o arquivo será classificado, exceto pelos vinte primeiros registros.

Também há a possibilidade de se selecionar apenas alguns registros iniciais do arquivo. Para isto, usar o cartão SORT com a opção 'STOPAFT' (*Stop After*).
Ex: SORT FILEDS=(...),STOPAFT=300
Neste exemplo somente os 300 primeiros registros serão classificados.

'SKIPREC' e 'STOPAFT' podem ser usados simultaneamente (ver página 16).
Ex: SORT FILEDS=(...),SKIPREC=10,STOPAFT=1000
Neste exemplo os 10 primeiros registros serão eliminados da classificação e os 1000 primeiros registros assim obtidos serão classificados.

Obs: as opções 'SKIPREC' e 'STOPAFT' são válidas para OPTION e MERGE, também.

COMO SELECIONAR OS REGISTROS DO ARQUIVO DE ENTRADA

1. Específicando o cartão de controle INCLUDE para os registros **necessários**
   ou
2. Específicando o cartão de controle OMIT para os registros **desnecessários**.

A escolha do cartão de controle INCLUDE ou OMIT depende de qual é mais fácil e eficiente para escrever, pois as funções destes cartões são opostas, e não podem ser usados ao mesmo tempo.

Selecionam-se os registros a incluir ou omitir, pela comparação do conteúdo do campo do registro com:

1. Outro campo do registro (por exemplo, pode-se selecionar registros cujo último nome do autor é igual ao último nome do instrutor);

ou

2. Uma constante, que pode ser um conjunto de caracteres, um número decimal ou um conjunto de caracteres hexadecimal (por exemplo, pode-se selecionar registros que tenham os caracteres 'HIST' no campo departamento).

Pode-se ter duas ou mais condições combinadas logicamente, utilizando os operadores AND e OR. Por exemplo, pode-se selecionar registros que contenham os caracteres 'HIST' ou 'PSYCH' no campo departamento.

O **DFSORT** segue as regras abaixo para preenchimento e truncamento :

1. na comparação campo com campo, o menor campo é preenchido adequadamente (com brancos ou zeros).
2. na comparação campo com constante, a constante é preenchida ou truncada dependendo do tamanho do campo. Constantes decimais são preenchidas ou truncadas à esquerda; constantes em caracter ou hexadecimal são preechidas ou truncadas à direita.

USO DO CARTÃO INCLUDE

Suponha que é fim de ano, e queira-se classificar por TÍTULO somente os livros necessários para o próximo ano. Se o número de exemplares vendidos neste ano de um particular livro é maior do que o número em stock, pode-se assumir que necessitam-se mais exemplares.

Para codificar o cartão INCLUDE que seleciona a condição acima, deve-se fazer o seguinte:
1. deixar pelo menos um branco e escreva INCLUDE
2. deixar pelo menos um branco e escreva COND=
3. escrever, entre parenteses, e separando por vírgulas:
   - a localização, o tamanho, e formato do dado do campo "número de livros vendidos".
   - o operador de comparação GT
   - a localização, o tamanho e o formato do dado do campo "número de livros em stock". Pode-se usar FORMAT= quando os campos envolvidos têm o mesmo formato) .

Feito o cartão INCLUDE, pode-se codificar o cartão SORT para classificar os registros selecionados.

Pode-se colocar o cartão SORT **antes** ou **depois** do cartão INCLUDE. Os cartões de controle não têm qualquer ordem específica; mas, como prática de uma boa documentação, coloque-os na ordem em que eles são processados (ver página 16).

Portanto os cartões INCLUDE e SORT podem ser codificados da seguinte forma:

```
1 2                                                                  71    80
                    -----------> NÚMERO DE LIVROS VENDIDOS NO ANO
                    |            -----------> NÚMERO DE LIVROS EM STOCK
                    |            |
                  -----        -----
    INCLUDE COND=(166,4,BI,GT,162,4,BI)
    SORT FIELDS=(1,75,CH,A)
                 -------
                    |
                    -------> TÍTULO DO LIVRO
```

Pode-se selecionar um dos seguinte operadores de comparação:

```
EQ .... IGUAL A
NE .... NÃO IGUAL A
GT .... MAIOR QUE
GE .... MAIOR OU IGUAL A
LT .... MENOR QUE
LE .... MENOR OU IGUAL A
```

Agora, suponha que se queira classificar somente os livros da editora 'COR', e que estejam sob a condição descrita anteriormente. Para isto, precisa-se que duas condições sejam verdadeiras:

1) o número de livros vendidos tem que ser maior do que o número de livros em stock, e
2) a editora do livro tem que ser 'COR'. Para adícionar esta segunda condição, deve-se expandir o cartão INCLUDE, descrito anteriormente, da seguinte forma:
   - acrescentar o operador lógico AND
   - comparar o conteúdo do campo editora com a constante 'COR'

Desta forma o SORT deverá receber os seguintes cartões de controle:

```
                      -----------> NÚMERO DE LIVROS VENDIDOS NO ANO
                      |           -----------> NÚMERO DE LIVROS EM STOCK
                      |           |
                    -----       -----
    INCLUDE COND=(166,4,BI,GT,162,4,BI,AND,106,4,CH,EQ,C'COR')
    SORT FIELDS=(1,75,CH,A)                 --------------------
                  -------                             |
                     |                                ---> SOMENTE EDITORA COR
                     -------> TÍTULO DO LIVRO
```

Obs: o campo "editora" tem 4 bytes, portanto a constante 'COR' será preenchida com um branco à direita.

Agora suponha que se queira classificar (em ordem descendemte) por título os livros dos cursos 00032 e 10347. Para esta classificação escrever os seguintes cartões de controle:

```
                            ---> SEL. CURSO No. 00032
                            |                    ----> SEL. CURSO No. 10347
                            |                    |
                     <------------------>     <------------------>
    INCLUDE COND=(115,5,CH,EQ,C'00032',OR,115,5,CH,EQ,C'10347')
```

ou

```
    INCLUDE COND=(115,5,CH,EQ,C'00032,10347')

    SORT FIELDS=(1,75,D),FORMAT=CH
                  ------
                    |
                    ---> TÍTULO DO LIVRO
```

Observar as regras de preenchimento e truncamento de campos constantes. No exemplo anterior, não se pode substituir C'00032' por C'32', porque constantes caracteres são preenchidas com brancos a direita.

USO DO CARTÃO OMIT

Suponha que se queira classificar, por título, todos os livros usados nos cursos, menos aqueles de leitura geral. Neste caso, pode-se usar o cartão de controle OMIT, que exclui aqueles registros que contém brancos no campo "departamento" do curso.

Pode-se colocar o cartão SORT **antes** ou **depois** do cartão OMIT. Os cartões de controle não têm qualquer ordem específica; mas, como prática de uma boa documentação, coloque-os na ordem em que eles são processados (ver página 16).

O formato do cartão de controle OMIT é o mesmo do cartão de controle INCLUDE. Então para excluir os livros de leitura geral:

```
                    ---> DEPARTAMENTO DO CURSO
                    |
                  -----
      OMIT COND=(110,5,CH,EQ,C' ')
      SORT FIELDS=(1,75,CH,A)
                  -----
                    |
                    ----> TÍTULO DO LIVRO
```

PARÂMETROS COMUNS A INCLUDE E OMIT

Tanto para INCLUDE quanto para OMIT podem-se utilizar as seguintes opções:

```
COND=ALL     COND=(ALL)
COND=NONE    COND=(NONE)
```

COMPARAÇÕES PERMITIDAS PARA INCLUDE E OMIT

O quadro abaixo mostra as comparações permitidas para campo com campo e campo com constante.

CAMPO com CAMPO						CAMPO com CONSTANTE		
	BI	**CH**	**ZD**	**PD**	**AC**	**CARACTER**	**HEXADECIMAL**	**DECIMAL**
BI	**X**	**X**				**X**	**X**	
CH	**X**	**X**				**X**	**X**	
ZD			**X**	**X**				**X**
PD			**X**	**X**				**X**
AC					**X**	**X**	**X**	

Por exemplo, para classificar por "último nome do autor" somente aqueles livros cujo "último nome do autor" comece por M, comparar o conteúdo do byte 76 (o primeiro byte do último nome do autor), que é formato caracter, com caracter ou caracter hexadecial, isto é:

```
        INCLUDE COND=(76,1,CH,EQ,C'M')
        SORT FIELDS=(76,15,CH,A)
    ou
        INCLUDE COND=(76,1,CH,EQ,X'D4')
```

```
        SORT FIELDS=(76,15,CH,A)
```

Para classificar por quantidade em stock somente os livros cuja quantidade em stock é menor do que 10, comparar o conteúdo do campo da quantidade em stock, que é formato binário, com uma constante hexadecimal, isto é :

```
                                ---> NÚMERO EM STOCK MENOR DO QUE 10
                                |
                    ------------------------
        INCLUDE COND=(162,4,BI,LT,X'0000000A')
        SORT FIELDS=(162,4,BI,A)
                    ----------
                        |
                        ---> NÚMERO EM STOCK
```

Lembre-se das regras de preenchimento de truncamento. Se se especificasse X'0A', esta constante seria preenchida à direita e não à esquerda.

COMO ESCREVER CONSTANTES

O formato para caracteres, caracteres hexadecimal e números decimal estão abaixo.

CONSTANTE CARACTER

O formato para escrever uma constante caracter é: **n**C'X...X'
   onde **n** é uma quantidade de 1-4095 (1 por default)
        X é um caracter EBCDIC (podem ser usados de 1 a 256 caracteres)
Ex: C'FERN', 5C'A'

Para incluir um único apóstrofe, específicar duas apostrofes.
Ex: O'NEILL deve ser específicado como C'O''NEILL'.

CONSTANTE HEXADECIMAL

O formato hexadecimal é: **n**X'YY...YY'
  onde **n** é uma quantidade de 1-4095 (1 por default)
       YY é um par de dígitos hexadecimais (podem ser usados de 1 a 256 pares)
Ex: 2X'0001'   ---> **2** x **PIC S9(4) COMP** com conteúdo 1
     X'0000000A' ---> **1** x **PIC S9(8) COMP** com conteúdo 10

CONSTANTE DECIMAL

O formato decimal é: N...N ou -N...N onde N é um dígito decimal.
Ex: 32, 52, AND -69.

Obs: o número decimal não pode conter vírgula ou ponto decimal.

Capítulo 4

SOMAR VALORES EM REGISTROS

Suponha que o departamento de ingles queira saber o preço total dos livros de todos os seus cursos. Para isto, pode-se selecionar todos os registros do departamento de ingles usando o cartão de controle INCLUDE, e somar os preços dos livros usando os cartões de controle SORT e SUM do **DFSORT**.

No cartão de controle SUM, especificar um ou mais campos numéricos que serão somados sempre que o conteúdo do campo refereciado pelo cartão de controle SORT for constante. O campo numérico pode ser binário, decimal compactado, ou decimal zonado.

Portanto para somar os preços de todos os registros do departamento de ingles, especificar o campo preço no cartão de controle SUM e o campo departamento é referenciado no cartão de controle SORT.

Quando somar conteúdo de campos de registros, dois campos são envolvidos:
1. Campos de controle, que são especificados no cartão de controle SORT e
2. Campos de sumário, que são especificados no cartão de controle SUM

O conteúdo dos campos de sumário, são somados somente quando o conteúdo do campos de controle são iguais.

USO DO CARTÃO SUM

Para escrever o cartão de controle SUM que some os preços dos livros do departamento de ingles:
- . Deixar pelo menos um branco e escrever SUM
- . Deixar pelo menos um branco e escrever FIELDS=
- . Escrever, entre parenteses, e separar por vírgulas:
  - .. a localização, o tamanho, e o formato do campo preço.

Os cartões de controle INCLUDE, SORT e SUM são da seguinte forma:

```
INCLUDE COND=(110,5,CH,EQ,C'ENGL') ---> SEL. DEP. DO CURSO INGLES
SORT FIELDS=(110,5,CH,A)           ---> DEPARTAMENTO DO CURSO
SUM FIELDS=(170,4,BI)              ---> PREÇO
```

Quando os preços são somados, o total é colocado no campo preço de um registro, e os demais são deletados.

Vamos supor agora que o departamento de ingles queira saber o preço total do livros de cada curso. Neste caso selecionar os registros do departamento de ingles usando INCLUDE, e específicar o campo preço no cartão SUM, mas deve-se específicar o campo número do curso no cartão SORT. Portanto, os cartões de controles são codificados da seguinte forma:

```
INCLUDE COND=(110,5,CH,EQ,C'ENGL') ---> SEL. DEP. DO CURSO INGLES
SORT FIELDS=(115,5,CH,A)           ---> NÚMERO DO CURSO
SUM FIELDS=(170,4,BI)              ---> PREÇO
```

Suponhamos agora que se queira somar separadamente o número de livros em stock e o número de livros vendidos por cada editora. Para isto específicar o campo editora no cartão de controle SORT, os campos número em stock e número de livros vendidos no cartão de controle SUM, isto é:

```
                        ---> EDITORA
                        |
                       -----
          SORT FIELDS=(106,4,CH,A)
          SUM FIELDS=(162,4,166,4),FORMAT=BI
                      ---   ---
                       |     |
                       |     ---> NÚMERO DE LIVROS VENDIDOS
                       ---> NÚMERO DE LIVROS EM STOCK
```

OVERFLOW

Se existe um campo binário de 2 bytes (sem sinal) contendo X'FFFF' e soma-se X'0001' a ele, ocorrerá *overflow*, porque o resultado requer mais do que 2 bytes, isto é:
X'FFFF' + X'0001' = X'010000'

Para o **DFSORT** este problema foi resolvido da seguinte forma: os dois registros envolvidos (dois registros são somados a cada vez) não serão somados, isto é, os seus conteúdos não serão destruídos, e nenhum dos registros será deletado. No entanto, outros registros continuarão sendo contabilizados. Se ocorrer *overflow* o **DFSORT** fornecerá uma mensagem como aviso.

Em alguns casos, pode-se corrigir o problema de *overflow* aumentando o campo sumário com zeros, usando o cartão de controle INREC (ver capítulo 6).

DELETAR REGISTROS COM CAMPO CONTENDO DADOS DUPLICADOS

Outra função do cartão SUM é deletar registros cujo campo de controle contém dados duplicados. Por exemplo, para se obter uma lista (em ordem ascendente) das editoras, com cada editora aparecendo uma única vez.

Se se específicar FIELDS=NONE no cartão de controle SUM e utilizar o cartão de controle SORT fazendo referência ao campo "editora", somente um registro de cada editora será classificado.

Codificar os seguintes cartões de controle:

```
          SORT FIELDS=(106,4,CH,A)  -----> CAMPO EDITORA
          SUM FIELDS=NONE
```

OPÇÃO ZDPRINT

A opção ZDPRINT permite a converção de campos positivos zonado-decimal em números printáveis, quando estes forem resultantes de uma operação ZD SUM.
Ex : o hexadecimal 'F3F2C5' é convertido para 325.

```
          OPTION ZDPRINT
          SORT FIELDS=(13,5,CH,A)
          SUM FIELDS=(36,5,ZD,50,5,ZD)
```

Capítulo 5

REFORMATANDO OS REGISTROS DE ENTRADA

Outra forma de reformatar os registros é utilizar o cartão de controle INREC. Com este cartão de controle, pode-se deletar campos, inserir brancos ou zeros, e reordenar campos.

A diferença entre os cartões de controle OUTREC e INREC é que: enquanto o OUTREC atua **depois** da classificação do registros, o INREC atua **antes** da classificação dos registros.

É óbvio que registros de tamanho menor gastam menos tempo para classificação; portanto, aconselha-se a utilização do cartão de controle INREC para deletar campos, e OUTREC para inserir brancos ou zeros. Ambos servem para reordenar campos.

A utilização do cartão de controle INREC ou OUTREC é indiferente para reorganizar os campos de um registro, supondo-se que o seu tamanho nunca se altere. Caso contrário, utilizar estes cartões de controle convenientemente.

USO DO CARTÃO INREC

O cartão de controle INREC tem o mesmo formato do cartão de controle OUTREC. Assim, no primeiro exemplo do capítulo 6, onde é usado o cartao OUTREC para obter somente os campos "editora", "número de livros em stock" e "número de livros vendidos", pode-se usar o cartão INREC da seguinte forma:

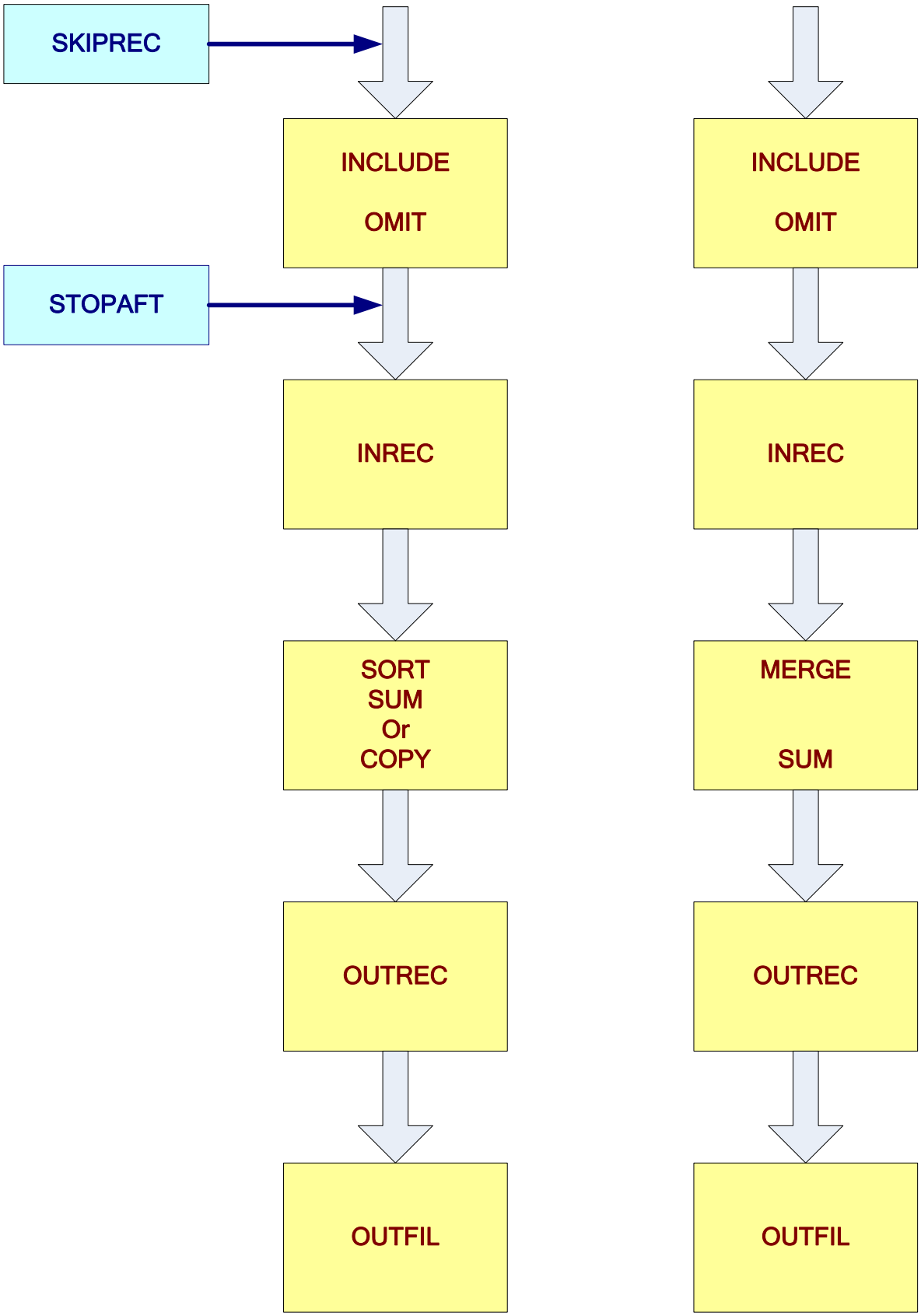
```
         INREC FIELDS=(106,4,162,4,166,4)
    ou
         INREC FIELDS=(106,4,162,8)
```

O cartão INREC reformata os registros antes da classificação, mas os cartões SORT e SUM devem fazer referência aos campos após a reformatação. Logo, a solução completa do exemplo anterior é:

```
        INREC FIELDS=(106,4,162,4,166,4)
        SORT FIELDS=(1,4,CH,A)
        SUM FIELDS=(5,4,BI,9,4,BI)
  ou
        INREC FIELDS=(106,4,162,8)
        SORT FIELDS=(1,4,CH,A)
        SUM FIELDS=(5,4,BI,9,4,BI)
```

O cartão INREC é processado antes dos cartões SORT, SUM e OUTREC, mas após os cartões INCLUDE e OMIT. Portanto, quando usar o cartão INREC, os cartões SORT, SUM, OUTREC e OUTFIL devem referenciar os campos reformatados, enquanto que os cartões INCLUDE e OMIT devem referenciar os campos nas suas posições originais.

Graficamente temos:



EVITANDO OVERFLOW QUANDO SOMAR VALORES EM REGISTROS

Em alguns casos, pode-se evitar *overflow* usando o cartão INREC para aumentar o campo de sumário com zeros. No entanto, este método não pode ser utilizado para campos em ponto fixo binário com valores negativos, porque o preenchimento com zeros à esquerda troca o sinal do campo.

No exemplo anterior pode-se prevenir o *overflow* nos campos sumário, se se adicionar mais 4 bytes à esquerda (campo binário deve ter 2, 4 ou 8 bytes de tamanho).

Portanto, os cartões de controle para o **DFSORT** são:

```
                        ---------------> INSERE 4 BYTES BINÁRIOS
                        |        |          PARA SEREM USADOS POR "SUM"
                        |        |
INREC FIELDS=(160,4,4Z,162,4,4Z,166,4)
SORT FIELDS=(1,4,CH,A)
SUM FIELDS=(5,8,BI,13,8,BI)
             ------ -------
               |      |
               |      ---> NOVO CAMPO PARA LIVROS VENDIDOS
               ---> NOVO CAMPO PARA LIVROS EM STOCK
```

FORMATAR CONDICIONALMENTE OS REGISTROS DE ENTRADA

O parâmetro FIELDS permite uma reformatação uniforme de todos os registros de entrada.
A cláusula IFTHEN do comando INREC permite uma reformatação condicional de diferentes sub-conjuntos de registros.

Existem 4 tipos de cláusulas IFTHEN:

1. WHEN=INIT: Use uma ou mais cláusulas WHEN=INIT para aplicar BUILD itens em todos os registros de entrada. Cláusulas WHEN=INIT são processadas antes qualquer outra cláusula IFTHEN.

2. WHEN=(logexp): Use uma ou mais cláusulas WHEN=(logexp) para aplicar BUILD itens ao sub-conjunto dos registros que satisfazem uma determinada expressão lógica. Pode-se usar qualquer expressão lógica utilizável no parâmetro COND=(logexp) de uma declaração INCLUDE.

3. WHEN=ANY: Use uma cláuisula WHEN=ANY após multiplas cláusulas WHEN=(logexp) para aplicar BUILD itens adicionais aos registros de entrada que satisfazem os critérios para qualquer das cláusulas WHEN=(logexp) precedentes.

4. WHEN=NONE: Use uma ou mais cláusulas WHEN=NONE para aplicar BUILD itens adicionais aos registros de entrada que não satisfazem nenhum dos critérios para qualquer das cláusulas WHEN=(logexp) precedentes. Cláusulas WHEN=NONE são processadas após qualquer todas as cláusulas IFTHEN. Se não for especificada nenhuma cláusula WHEN=NONE, somente as cláusulas WHEN=INIT (se existentes) são aplicadas aos registros de entrada que não satisfazem nenhum dos critérios para qualquer das cláusulas WHEN=(logexp) precedentes.

Ex 1:

```
INREC
IFTHEN=(WHEN=INIT,BUILD=(1,100,50X)),
IFTHEN=(WHEN=(50,4,CH,EQ,C'COR',AND,70,4,BI,GT,+2000),BUILD=(1,100,50X)),
IFTHEN=(WHEN=(50,4,CH,EQ,C'COR',AND,70,4,BI,LE,+2000),BUILD=(1,090,60X)),
IFTHEN=(WHEN=ANY,BUILD=(1,100,50X))
IFTHEN=(WHEN=NONE,BUILD=(1,100,50X))
```

Ex 2:

```
//SYMNAMES DD *
TIPO_CLI,15,01,CH
/*

//SYSIN DD *
  INREC
  IFTHEN=(WHEN=(TIPO_CLI,EQ,C'1'),BUILD=(001,014,C'F',016,050)),
  IFTHEN=(WHEN=(TIPO_CLI,EQ,C'2'),BUILD=(001,014,C'J',016,050)),
  IFTHEN=(WHEN=NONE,BUILD=(001,014,C'Z',016,050))
  SORT FIELDS=(001,015,CH,A)
/*
```

Obs: IFTHEN também pode ser usado com OUTREC.

Capítulo 6

REFORMATANDO OS REGISTROS DE SAÍDA

Após a classificação dos registros e antes deles serem escritos no arquivo de saída, pode-se reformatá-los com o uso do cartão de controle OUTREC.

Utilizando-se o cartão de controle OUTREC pode-se:
- . Eliminar campos
- . Rearranjar a ordem dos campos
- . Inserir zeros antes, entre ou após os campos
- . Inserir brancos antes, entre ou após os campos
- . Converter campos numéricos para vários formatos
- . Formatar condicionalmente os registros de saída

Obs: se se usar o cartão de controle OUTREC para modificar o tamanho do registro (pela deleção de campos ou inserção de brancos ou zeros), obviamente será necessário especificar um novo tamanho de registro no cartão DDNAME SORTOUT, via parâmetro DCB.

USO DO CARTÃO OUTREC

DELETAR CAMPOS

No capítulo anterior, foi usado o cartão de controle SUM para somar todos os livros em stock e todos os livros vendidos por cada editora.

Agora, utilizando o cartão de controle OUTREC, pode-se deletar os campos que não são necessários para a aplicação. Somente os campos "editora", "número de livros em stock" e "número de livros vendidos" serão úteis, com isto cada registro do arquivo de saída ficará apenas com 12 bytes.

Para escrever o cartão de controle OUTREC:
- . Deixar pelo menos um branco e escreve OUTREC
- . Deixar pelo menos um branco e escreve FIELDS=
- . Escrever, entre parenteses, e separando por vírgulas:
  - .. a localização e tamanho do campo editora
  - .. a localização e tamanho do campo livros em stock
  - .. a localização e tamanho do campo livros vendidos

Visto que os campos "livros em stock" e "livros vendidos" são contíguos, pode-se especificá-los como um único campo (eles não precisam ter o mesmo formato de dados).

Observe que neste cartão de controle não se especifica o formato do dado. Portanto, a codificação dos cartões de controle para o **DFSORT** fica:

```
SORT FIELDS=(106,4,CH,A)
SUM FIELDS=(162,4,BI,166,4,BI)
OUTREC FIELDS=(106,4,162,4,166,4)
               ----- ----- -----
                 |     |     |
                 |     |     ---> CAMPO LIVROS VENDIDOS
                 |     ---> CAMPO LIVROS EM STOCK
                 ---> CAMPO EDITORA
```

ou

```
SORT FIELDS=(106,4,CH,A)
SUM FIELDS=(162,4,BI,166,4,BI)
OUTREC FIELDS=(106,4,162,8)
               ----- -----
                 |     |
                 |     ---> LIVROS EM STOCK e LIVROS VENDIDOS
                 ---> EDITORA
```

Conforme foi dito, teremos um novo tamanho de registro no arquivo de saída; neste caso o tamanho de registro é 12 bytes. Portanto teremos que específicar o parâmetro DCB para o arquivo de saída do **DFSORT**:

```
//SORTOUT  DD  DSN=BOOKS.OUTPUT,DISP=(NEW,CATLG),UNIT=SYSDA,
//         SPACE=(12000,20),UNIT=SYSDA,
//         DCB=(LRECL=12,BLKSIZE=12000,RECFM=FB)
               --------
                  |
                  ---> NOVO TAMANHO DE REGISTRO
```

REORDENAR CAMPOS

Os campos, dentro de um registro, sempre aparecem na mesma ordem em que são especificados. Então, se se deseja que o campo "livros vendidos" apareça antes do campo "livros em stock", reverter a ordem de especificação dos campos no cartão de controle OUTREC:

```
SORT FIELDS=(106,4,CH,A)
SUM FIELDS=(162,4,166,4),FORMAT=BI
OUTREC FIELDS=(106,4,166,4,162,4)
                      -----------
                           |
                           ---------> INVERSÃO DOS CAMPOS
```

INSERIR ZEROS BINÁRIOS

No exemplo construído anteriormente, assuma que se queira reformatar os registros, incluindo mais um campo de 4 bytes binário após o campo "livros em stock" (começando no byte 13). Neste caso, pode-se inserir zeros binários prevendo um uso futuro.

Para inserir os zeros, escrever 4Z após o último campo, conforme abaixo:

```
SORT FIELDS=(106,4,CH,A)
SUM FIELDS=(162,4,BI,166,4,BI)
OUTREC FIELDS=(106,4,166,4,162,4,4Z)
                                  |
                                  ---> INSERE 4 ZEROS BINÁRIOS
```

Deve-se especificar um novo tamanho de registro para o arquivo de saída:

```
//SORTOUT   DD  DSN=BOOKS.OUTPUT,DISP=(,CATLG),
//          SPACE=(16000,20),UNIT=SYSDA,
//          DCB=(LRECL=16,BLKSIZE=16000,RECFM=FB)
               --------
                  |
                  ---> NOVO TAMANHO DE REGISTRO
```

INSERIR BRANCOS

Se o arquivo de saída contém somente caracteres, pode-se imprimí-lo. Para isto basta fazer o sequinte:

```
//SORTOUT  DD  SYSOUT=A
```

Pode-se imprimir somente alguns campos legíveis do arquivo, usando o cartão de controle OUTREC, separando os campos envolvidos com brancos.

Suponha que se queira imprimir somente o campo "editora" e o campo "título", mas o campo "editora" deve aparecer primeiro. Esta impressão pode ser mais legível se se separar os dois campos com 10 brancos e criar uma margem de 20 brancos.

Para inserir brancos, escrever 10X entre os dois campos e 20X antes do primeiro campo. O cartão de controle SORT classificará os registros por título em ordem ascendente.
Obs: o cartão de controle SORT ou MERGE é sempre necessário.

A codificação dos cartões de controle para o **DFSORT** é:

```
          SORT FIELDS=(1,75,CH,A)
          OUTREC FIELDS=(20X,106,4,10X,1,75)
                         |          |
                         |          -----> 10 BRANCOS
                         -----> 20 BRANCOS
```

INSERIR VALORES HEXADECIMAIS

Para inserir valores diferentes de zeros, podemos utilizar constantes hexadecimais (ver pág. 12)

```
         SORT FIELDS=(106,4,CH,A)
         OUTREC FIELDS=(106,4,166,4,162,4,2X'00000001')
                                          |
                                          ---> INSERE 2 CAMPOS "PIC S9(8) COMP"
                                               COM CONTEÚDO 1
```

Deve-se especificar um novo tamanho de registro para o arquivo de saída:

```
//SORTOUT   DD  DSN=BOOKS.OUTPUT,DISP=(,CATLG),
//          SPACE=(16000,20),UNIT=SYSDA,
//          DCB=(LRECL=20,BLKSIZE=20000,RECFM=FB)
                --------
                   |
                   ---> NOVO TAMANHO DE REGISTRO
```

INSERIR VALORES CARACTERES

Para tanto, podemos utilizar constantes caractes (ver pág. 12)

```
         SORT FIELDS=(1,4,CH,A)
         OUTREC FIELDS=(1,100,C'ARQ00001')
                              -----------
                                   |
                                   ---> INSERE 1 CAMPO "PIC X(8)"
                                         COM CONTEÚDO 'ARQ00001'
```

Deve-se especificar um novo tamanho de registro para o arquivo de saída:

```
//SORTOUT   DD  DSN=ARQ000001.OUTPUT,DISP=(,CATLG),
//          SPACE=(11000,20),UNIT=SYSDA,
//          DCB=(LRECL=108,RECFM=FB)
                --------
                   |
                   ---> NOVO TAMANHO DE REGISTRO
```

CONVERTER CAMPOS NUMÉRICOS PARA DIVERSOS FORMATOS

Para converter valores numéricos de um formato para outro, utilizar a opção **TO** do cartão de controle OUTREC.
Ex:

```
         OUTREC FIELDS=(1,4,
                        X,
                        5,5,PD,TO=ZD,LENGTH=11,
                        X,
                        10,5,PD,TO=ZD,LENGTH=7)
```

```
    5   10   15   20   25   30
----+----+----+----+----+----+----+----+----+----+----+----+----+----+
AAAA 00000001234 0000111
BBBB 00000006789 0000222
CCCC 00000004567 0000333
```

Os cartões de controle acima realizam as seguintes conversões:
1. O campo compactado que ocupa as posições de 5 a 9 para um campo zonado de tamanho 11 (deveria ser 9);
2. O campo compactado que ocupa as posições de 10 a 14 para um campo zonado de tamanho 7 (deveria ser 9).

O parâmetro LENGTH, se usado, pode alterar o tamanho padrão para a conversão.

Vários formatos numéricos podem ser convertidos entre si; os mais comuns são: BI, PD e ZD.

EDITAR CAMPOS NUMÉRICOS

Para editar campos no formato PD (decimal compactado) para formato caracter, pode-se usar as capacidades de edição do comando OUTREC, além de inserir sinais, vírgulas, pontos e hífens.

Dado um registro com 3 campos decimais compactados onde
. o 1°. campo possui 2 decimais
. o 2o. campo possui 3 decimais
. o 3o. campo é uma data no formato mmddyyyy

Abaixo um exemplo de conteúdo legível (na realidade não o é):

Campo caracter	**1°. campo PD**	**2°. campo PD**	**3°. campo PD**
1 8	11 16	31 37	41 45
WEST	+1524900810	+0000000020000	+05122003
EAST	-0065781053	+3721500532006	+11292003
NORTH	+0000000000	-0000982630735	+02152004
SOUTH	-0000003562	-0003826254999	+12032003

```
OUTREC FIELDS=(1,8,
               5X,
               11,6,PD,EDIT=(SII,III,III,III,III,III,III,III,III,IIT.TT),
               5X,
               31,7,PD,EDIT=(SI,III,III,IIT.TTT),SIGNS=(,-),
               5X,
               41,5,PD,EDIT=(TT-TT-TTTT))
```

I indica que os dígitos iniciais
   diferentes de zero devem ser mostrados tais quais
   iguais   a zero devem ser mostrados como brancos

T indica que os dígitos a que se refere devem ser mostrados tais quais

S no começo ou no fim indica um sinal + ou -

SIGNS=(,-) altera o comportamento de **S** indicando que sinais positivos devem ser mostrados como brancos

Qualquer outro caracter (vírgula, ponto ou hífen) é mostrado tal qual.

O resultado será:

```
WEST    +15,249,008.10                20.000    05-12-2003
EAST       -657,810.53    3,721,500,532.006    11-29-2003
NORTH            +0.00         -982,630.735    02-15-2004
SOUTH           -35.62       -3,826,254.999    12-03-2003
```

FORMATAR CONDICIONALMENTE OS REGISTROS DE SAÍDA

O parâmetro FIELDS permite uma reformatação uniforme de todos os registros de saída. A cláusula IFTHEN do comando OUTREC permite uma reformatação condicional de diferentes sub-conjuntos de registros.

Existem 4 tipos de cláusulas IFTHEN:

1. WHEN=INIT: Use uma ou mais cláusulas WHEN=INIT para aplicar BUILD items em todos os registros de entrada. Cláusulas WHEN=INIT são processadas antes qualquer outra cláusula IFTHEN.

2. WHEN=(logexp): Use uma ou mais cláusulas WHEN=(logexp) para aplicar BUILD itens ao sub-conjunto dos registros que satisfazem uma determinada expressão lógica. Pode-se usar qualquer expressão lógica utilizável no parâmetro COND=(logexp) de uma declaração INCLUDE.

3. WHEN=ANY: Use uma cláuisula WHEN=ANY após multiplas cláusulas WHEN=(logexp) para aplicar BUILD itens adicionais aos registros de entrada que satisfazem os critérios para qualquer das cláusulas WHEN=(logexp) precedentes.

4. WHEN=NONE: Use uma ou mais cláusulas WHEN=NONE para aplicar BUILD itens adicionais aos registros de entrada que não satisfazem nenhum dos critérios para qualquer das cláusulas WHEN=(logexp) precedentes. Cláusulas WHEN=NONE são processadas após qualquer todas as cláusulas IFTHEN. Se não for especificada nenhuma cláusula WHEN=NONE, somente as cláusulas WHEN=INIT (se existentes) são aplicadas aos registros de entrada que não satisfazem nenhum dos critérios para qualquer das cláusulas WHEN=(logexp) precedentes.

Ex:

```
OUTREC
IFTHEN=(WHEN=INIT,BUILD=(1,100,50X)),
IFTHEN=(WHEN=(50,4,CH,EQ,C'COR',AND,70,4,BI,GT,+2000),BUILD=(1,100,50X)),
IFTHEN=(WHEN=(50,4,CH,EQ,C'COR',AND,70,4,BI,LE,+2000),BUILD=(1,100,50X)),
IFTHEN=(WHEN=ANY,BUILD=(1,100,50X))
IFTHEN=(WHEN=NONE,BUILD=(1,100,50X))
```

Obs: IFTHEN também pode ser usado com INREC.

ALIGNAR UM CAMPO À DIREITA OU À ESQUERDA

As cláusulas BUILD, OVERLAY, IFTHEN BUILD e IFTHEN OVERLAY dos comandos INREC, OUTREC e OUTFIL permitem o uso da opção JFY (justify) para alignar à esquerda ou à direita o conteúdo de um campo.

Para alinhar à esquerda, os brancos iniciais são removidos e todos os caracteres desde o primeiro não branco até o último não branco são movidos para a esquerda, com brancos sendo inseridos à direita se necessário.

Para alinhar à direita, os brancos finais são removidos e todos os caracteres desde o último não branco até o primeiro não branco são movidos para a direita, com brancos sendo inseridos à esquerda se necessário.

Ex:

Suponha registros como os abaixo:

```
      History
  Psychology
         Business
        Biology
   Computer Science
```

Para ficar mais apresentável, podemos alinhar à esquerda as posições de 1 a 30:

```
  OPTION COPY
  OUTREC FIELDS=(1,30,JFY=(SHIFT=LEFT))
```

```
History
Psychology
Business
Biology
Computer Science
```

Ou podemos alinhar à direita estas mesmas posições:

```
  OPTION COPY
  OUTREC FIELDS=(1,30,JFY=(SHIFT=RIGHT))
```

```
                     History
                  Psychology
                    Business
                     Biology
            Computer Science
```

COMPRIMIR UM CAMPO À DIREITA OU À ESQUERDA

As cláusulas BUILD, OVERLAY, IFTHEN BUILD e IFTHEN OVERLAY dos comandos INREC, OUTREC e OUTFIL permitem o uso da opção SQZ (squeeze) para comprimir à esquerda ou à direita o conteúdo de um campo.

Para comprimir à esquerda, todos os caracteres não brancos são removidos e os caracteres desde o primeiro não branco até o último não branco são movidos para a esquerda, com brancos sendo inseridos à direita se necessário.

Para comprimir à direita, todos os caracteres não brancos são removidos e os caracteres desde o último não branco até o promeiro não branco são movidos para a direita, com brancos sendo inseridos à esquerda se necessário.

Ex:

Suponha registros como os abaixo:

```
<tag>    History   </tag>
  <tag>     Psychology     </tag>
 <tag>  Business  </tag>
    <tag>Biology </tag>
  <tag> Science   </tag>
```

Para remover os brancos, podemos comprimir para a esquerda os 40 primeiros caracteres:

```
  OPTION COPY
  OUTREC FIELDS=(1,40,SQZ=(SHIFT=LEFT))
```

com o seguinte resultado:

```
<tag>History</tag>
<tag>Psychology</tag>
<tag>Business</tag>
<tag>Biology</tag>
<tag>Science</tag>
```

Ou podemos comprimir à direita estas mesmas posições:

```
  OPTION COPY
  OUTREC FIELDS=(1,40,SQZ=(SHIFT=RIGHT))
```

```
                   <tag>History</tag>
                <tag>Psychology</tag>
                  <tag>Business</tag>
                   <tag>Biology</tag>
                   <tag>Science</tag>
```

Capítulo 7

FUSÃO DE ARQUIVOS

Geralmente, a razão para *merge* de arquivos é adicionar mais registros ao arquivo que já está classificado. Por exemplo, se o arquivo *BOOKSTORE* já estiver classificado por departamento e título dos livros, e quizermos atualizá-lo, via *merge*, usando um arquivo que contém 5 registros, também classificado por departamento e título de livros, pode-se utilizar o cartão de controle MERGE do **DFSORT**.

USO DO CARTÃO MERGE

Para fundir arquivos, deve-se escrever o cartão de controle MERGE e vários cartões de JCL. Sempre que se fundem arquivos, estes arquivos devem estar classificados pelos mesmos campos e os registros devem ter o mesmo formato. Pode-se fundir até 16 arquivos simultaneamente.

O formato do cartão MERGE é igual ao do cartão SORT e todos os cartões descritos anteriormente (INCLUDE, OMIT, SUM, OUTREC e INREC) são válidos para a função MERGE, bem como o cartão OUTFIL.

Então, para atualizar o arquivo *BOOKSTORE* usando o novo arquivo que contém 5 registros, escrever o sequinte cartão MERGE:

```
MERGE FIELDS=(110,5,A,1,75,A),FORMAT=CH
              ------- ------
                 |       |
                 |       ------------> DEPARTAMENTO
                 -------------------> TÍTULO DOS LIVROS
```

JCL PARA O MERGE

Os cartões de JCL para o MERGE são os mesmos que são utilizados para SORT com as seguintes exceções:

- Não usar cartão DD SORTWKnn.
- Ao invés de SORTIN DD, usar SORTINxx DD para definir os arquivos de entrada (um SORTINxx para cada arquivo que será fundido; xx deve ser um número de 00 até 99).

Para fundir o arquivo *BOOKSTORE* com o arquivo que contém os novos registros, codificar o seguinte JCL: (assumir que os arquivos de entrada estejam catalogados e que o arquivo de saída será catalogado)

```
//EXAMP    JOB  A492,PROGRAMMER
//SORT     EXEC PGM=SORT,REGION=512K
//SYSOUT    DD  SYSOUT=*
//SORTIN01 DD  DSN=BOOKS.INPUT1,DISP=OLD
//SORTIN02 DD  DSN=BOOKS.INPUT2,DISP=OLD
//SORTOUT  DD  DSN=BOOKS.OUTPUT,DISP=(,CATLG),UNIT=SYSDA,
//         SPACE=(CYL,(1,1))
//SYSIN    DD  *
        MERGE FIELDS=(110,5,A,1,75,A),FORMAT=CH
/*
```

Capítulo 8

CHAMANDO O DFSORT VIA PROGRAMA

Pode-se chamar o **DFSORT** via programas escritos em Cobol, PL1 ou Assembler. Veremos apenas o SORT e MERGE usando as linguagens Cobol e PL1.

PASSANDO CARTÃO DE CONTROLE DO DFSORT

Quando se está usando Cobol ou PL1, é possível passar os cartões de controle INCLUDE, OMIT, SUM, INREC e/ou OUTREC (estes programas criam os cartões de controle RECORD, SORT ou MERGE) para o **DFSORT** via JCL utilizando-se o cartão DD SORTCNTL. Por exemplo, pode-se passar o cartão de controle INCLUDE que selecionará somente os livros do departamento de ingles da seguinte forma:

```
//BOOKS  EXEC PGM=COBOLPGM
//.
//SORTCNTL   DD  *
     INCLUDE COND=(110,5,CH,EQ,C'ENGL')
/*
```

CHAMANDO O DFSORT VIA COBOL

Para chamar o **DFSORT** via programa em Cobol, usar os comandos SORT ou MERGE do Cobol.

CLASSIFICANDO REGISTROS VIA COBOL

O exemplo abaixo mostra como chamar o SORT para classificar o arquivo *BOOKSTORE* (MASTER-FILE) por título do livro. O arquivo classificado é gravado no arquivo SORTED-MASTER-FILE. Note que o campo e a ordem para classificação é fornecida pelo Cobol.

```
           SORT-ROUTINE SECTION.
               SORT SD-FILE
               ASCENDING KEY TITLE-IN USING MASTER-FILE
               GIVING SORTED-MATER-FILE.
                     .
                     .
               IF SORT-RETURN > 0 DISPLAY "ERRO NO SORT".
```

USANDO MERGE DE REGISTROS VIA COBOL

O exemplo abaixo mostra como chamar o **DFSORT** para fundir o arquivo *BOOKSTORE* pré-classificado com outro arquivo pré-classificado, para criar o novo arquivo.

```
          MERGE-ROUTINE SECTION.
               MERGE SD-FILE
               ASCENDING KEY TITLE-KEY USING MASTER-FILE NEW-BOOKS-FILE
               GIVING MERGED-FILE.
                     .
                     .
               IF SORT-RETURN > 0 DISPLAY "ERRO NO MERGE".
```

CHAMANDO O DFSORT VIA PL/I

Quando chamar o **DFSORT** via PL1, as informações a passar para o **DFSORT** incluem um cartão SORT ou MERGE. No cartão de controle RECORD, específicar o tipo e o tamanho dos registros. Seguindo este cartão deve-se fornecer a quantidade de memória em bytes.

Pode-se passar outros cartões de controle do **DFSORT** via cartão DD SORTCNTL.

O exemplo abaixo mostra como chamar o **DFSORT** via PL1.

```
        CALL PLISRTA (' SORT FIELDS=(1,75,CH,A) ',
                      ' RECORD TYPE=F,LENGTH=(173) ', 900000,
                      RETURN_CODE);

        IF RETURN_CODE ^= 0 THEN DO ;
           PUT SKIP EDIT ('ERRO NO SORT')(A);
           CALL PLIRETC(RETURN_CODE);
        END;
```

JCL PARA EXECUTAR UM PROGRAMA EM PL1 QUE USA O DFSORT

```
//EXAMP    JOB  A492,PROGRAMMER
//BOOKS    EXEC PGM=PLIPGM
//STEPLIB  DD  DNS=USER.PGMLIB,DISP=SHR
//SYSOUT   DD  SYSOUT=*
//SORTIN   DD  DSN=BOOKS.INPUT,DISP=OLD
//SORTWK01 DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SORTCNTL DD  *
     INCLUDE COND=(110,5,CH,EQ,C'ENGL')
/*
//SYSPRINT DD  SYSOUT=*
```

Obs: O cartão SORTCNTL foi utilizado para passar ao **DFSORT** o cartão de controle INCLUDE, o qual selecionará somente os livros do departamento de ingles.

Capítulo 9

ALTERANDO PARÂMETROS DEFAULT DO DFSORT

Quando o **DFSORT** é instalado, ele possui parâmetros *default* que podem ser alterados, dependendo das necessidades. Isto pode ser feito via JCL ou utilizando-se o cartão de controle OPTION do **DFSORT**.

ALTERANDO PARÂMETROS DEFAULT VIA JCL

Se se está executando o **DFSORT** via JCL, pode-se usar o parâmetro PARM do JCL para alterar certos *default*. Por exemplo, se o *default* da instalação é listar os cartões de controle do **DFSORT** e não se quer que eles sejam listados, pode-se especificar NOLIST como parâmetro via JCL:

```
//SORT  EXEC PGM=SORT,PARM='NOLIST'
```

ALTERANDO PARÂMETROS DEFAULT VIA CARTÃO OPTION

Se se chama o **DFSORT** via programa ou executa-o via JCL, pode-se utilizar o cartão de controle OPTION para alterar certos *defaults*. Para isto, colocar o cartão OPTION entre outros cartões do **DFSORT**, os quais devem seguir o cartão 'DD SYSIN' ou 'DD SORTCNTL'.

Uma opção que pode ser alterada segundo a necessidade é a opção de CHECKPOINT/RESTART, pois em nossa instalação o **DFSORT** tem como *default* não tomar checkpoint/restart; visto que a melhor técnica de classificação do **DFSORT** (*blockset*) não suporta CHECKPOINT/RESTART.

Portanto, se a técnica *blockset* for escolhida pelo **DFSORT**, não teremos CHECKPOINT/RESTART (opção de instalação). No entanto, se for necessário tomar CHECKPOINT/RESTART, a técnica *blockset* tem que ser ignorada de qualquer forma. Para que isto ocorra, deve-se usar o cartão OPTION especificando NOBLKSET e, obviamente, solicitar a tomada de CHECKPOINT/RESTART, isto é:

```
          SORT FIELDS=(1,75,CH,A),CKPT
          OPTION NOBLKSET
    ou
          SORT FIELDS=(1,75,CH,A)
          OPTION NOBLKSET,CKPT
```

Convém observar que algumas opções DEFAULT podem ser alteradas via cartão OPTION, mas não via JCL, e outras opções podem ser alteradas via JCL, mas não via cartão OPTION.

Capítulo 10

COPIANDO ARQUIVOS

Com o **DFSORT** pode-se copiar dados sem executar a operação de classificação ou fusão. À exceção da operação SUM, pode-se usar qualquer cartão de controle discutido anteriormente. Em outras palavras, o **DFSORT** dá a opção de selecionar ou reformatar os registros quando se está copiando um arquivo.

Pode-se específicar a opção COPY no cartão SORT, MERGE, ou no cartão de controle OPTION.

Todos os cartões descritos anteriormente (INCLUDE, OMIT, SUM, OUTREC e INREC) são válidos para a função COPY.

COMO SOLICITAR A FUNÇÃO COPY DO DFSORT

O cartão de controle SORT ou MERGE devem ser codificados da seguinte forma:

```
SORT  FIELDS=COPY   ou
MERGE FIELDS=COPY
```

Se se quizer utilizar o cartão OPTION ele deve ser codificado da seguinte maneira:

```
OPTION COPY
```

USANDO A FUNÇÃO COPY COM OS CARTÕES DE CONTROLE INCLUDE E INREC

Para se copiar o arquivo *BOOKSTORE*, que já está ordenado por departamento e título, somente com os registros do departamento "COMP", sem os campos "preços", utilizar:

1. o cartão de controle INCLUDE para selecionar o departamento 'COMP';
2. o cartão de controle INREC para eliminar os campos "preços";
3. o cartão de controle OPTION para específicar a função COPY.

```
INCLUDE COND=(110,5,CH,EQ,C'COMP')
INREC FIELDS=(1,114)
OPTION COPY
```

JCL PARA COPIAR O ARQUIVO

Os cartões de JCL para utilizar a função COPY do **DFSORT** são os mesmos usados para as funções SORT ou MERGE com uma exceção:

NÃO SE USA OS CARTÕES SORTWKnn

```
//EXAMP   JOB  A492,PROGRAMMER
//SORT    EXEC PGM=SORT,REGION=512K
//SYSOUT   DD  SYSOUT=*
//SORTIN   DD  DSN=BOOKS.INPUT,DISP=OLD
//SYSIN    DD  *
       INCLUDE COND=(110,5,CH,EQ,C'COMP')
       INREC FIELDS=(1,114)
       OPTION COPY
/*
//SORTOUT  DD  DSN=BOOKS.OUTPUT,DISP=(,CATLG),
//         SPACE=(1140,20),UNIT=SYSDA,
//         DCB=(LRECL=114,BLKSIZE=1140,RECFM=FB)
```

Obs: pode-se usar 'SORT FIELDS=COPY' ou 'MERGE FIELDS=COPY' ao invés de
'OPTION COPY' para produzir o mesmo resultado.

Se não se quizer alguns registros iniciais do seu arquivo de entrada, pode-se acrescentar ao cartão de contole OPTION a opção 'SKIPREC' (*Skip Records*);
Ex: OPTION COPY,SKIPREC=2
Neste exemplo o arquivo será copiado, exceto pelos dois primeiros registros.

Também há a possibilidade de se copiar apenas alguns registros iniciais do arquivo. Para isto usar o cartão OPTION com a opção 'STOPAFT' (*Stop After*).
Ex: OPTION COPY,STOPAFT=30
Neste exemplo somente 30 registros serão copiados.

'SKIPREC' e 'STOPAFT' podem ser usados simultaneamente (ver página 16).
Ex: OPTION COPY,SKIPREC=10,STOPAFT=50
Neste exemplo os 10 primeiros registros serão eliminados da cópia e os 50 primeiros registros assim obtidos serão copiados.

Obs: as opções 'SKIPREC' e 'STOPAFT' são válidas para MERGE OU SORT, também.

PROGRAMA ICEGENER

Este programa também pode ser utilizado para copiar arquivos seqüenciais e é equivalente ao utilitário IEBGENER.

Exemplo de JCL para a execução do ICEGENER:

```
//STEP     EXEC PGM=ICEGENER
//SYSPRINT  DD  SYSOUT=*
//SYSUT1    DD  DSN=FILE.INPUT,DISP=OLD
//SYSUT2    DD  DSN=FILE.OUTPUT,DISP=OLD
//SYSIN     DD  DUMMY
```

Observar que o JCL de execução do programa ICEGENER é exatamente igual ao do utilitário IEBGENER, exceto pelo cartão EXEC que deverá ser codificado EXEC PGM=ICEGENER. Agora, se o cartão DD SYSIN não for DUMMY, o programa ICEGENER chamará o IEBGENER para a realização das funções requisitas.

Capítulo 11

GERANDO VÁRIOS ARQUIVOS DE SAÍDA

Normalmente o **DFSORT** gera apenas um arquivo de saída (SORTOUT), mas o cartão de controle OUTFIL permite a geração de um ou mais arquivos de saída, cada um gerado pela aplicação do conjunto de parâmetros do cartão OUTFIL aos *datasets* de entrada (SORTIN); todos os arquivos gerados pelo mesmo cartão OUTFIL são idênticos (se não for especificado o parâmetro SPLIT ou SPLITBY). Vários cartões de controle OUTFIL podem ser utilizados simultaneamente.

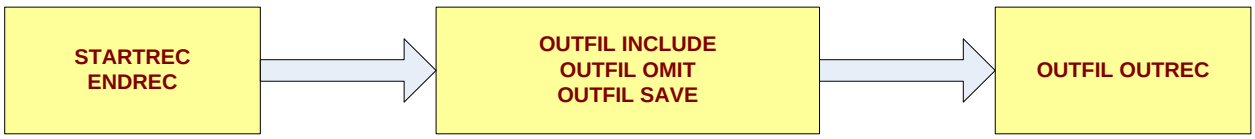
OUTFIL pode ser usado com SORT, MERGE ou COPY (ver pág.16).

OUTFIL é o último cartão de controle a ser executado e permite as seguintes tarefas:
. criação de vários *datasets* de saída, editados ou não, a partir de um ou mais *datasets* de entrada
. cada *dataset* de saída pode conter diferentes intervalos ou sub-conjuntos de registros, a partir dos *datasets* de entrada
. registros que não foram selecionadas para nenhum *dataset* de saída podem ser salvos num *dataset* específico
. conversão de registros de tamanho variável em registros de tamanho fixo
. os registros dos *datasets* de entrada podem ser gravados, alternadamente, entre os diversos *datasets* de saída

A sintaxe simplificada é a seguinte:

```
OUTFIL  FNAMES=(ddname1,...,ddnameN)
        FILES=(sufixo1,....,sufixoN)
        STARTREC=N
        ENDREC=N
        INCLUDE=...
        OMIT=...
        SAVE
        OUTREC=(campo1,....,campoN)
        CONVERT
        SPLIT
        SPLITBY
```

A ordem de execução dos parâmetros do OUTFIL é a seguinte:



11.1 FNAMES

Especifica os *ddnames* associados aos *datasets* de saída, os quais são carregados ao se aplicar os outros parâmetros do cartão OUTFIL aos *datasets* de entrada (SORTIN).

```
//SORT    EXEC PGM=SORT
//SYSOUT   DD  SYSOUT=*
//SORTIN   DD  DSN=BOOKS.INPUT,DISP=OLD
//SYSIN    DD  *
     OUTFIL FNAMES=(OUT1,OUT2),...
/*
//OUT1     DD  DSN=BOOKS.OUTPUT1,DISP=(,CATLG), ...
//OUT2     DD  DSN=BOOKS.OUTPUT2,DISP=(,CATLG), ...
```

11.2 FILES

Especifica os sufixos para os *ddnames* associados aos *datasets* de saída, os quais são carregados ao se aplicar os outros parâmetros do cartão OUTFIL aos *datasets* de entrada (SORTIN).

Os sufixos podem ser: 1 ou 2 carateres alfanuméricos para associar com os *ddnames* SORTOF**n** ou SORTOF**nn**, ou OUT para associar com SORTOUT.

Se nem FNAMES nem FILES são usados, o *default* para o *dataset* de saída é SORTOUT.

```
//SORT    EXEC PGM=SORT
//SYSOUT   DD  SYSOUT=*
//SORTIN   DD  DSN=BOOKS.INPUT,DISP=OLD
//SYSIN    DD  *
     OUTFIL FILES=(1,2,X),...
     OUTFIL FILES=(10,AA,BB),...
/*
//SORTOF1     DD  DSN=BOOKS.OUTPUT1,DISP=(,CATLG)
//SORTOF2     DD  DSN=BOOKS.OUTPUT2,DISP=(,CATLG)
//SORTOFX     DD  DSN=BOOKS.OUTPUTX,DISP=(,CATLG)
//SORTOF10    DD  DSN=BOOKS.OUTPUT10,DISP=(,CATLG)
//SORTOFAA    DD  DSN=BOOKS.OUTPUTAA,DISP=(,CATLG)
//SORTOFBB    DD  DSN=BOOKS.OUTPUTBB,DISP=(,CATLG)
```

11.3 STARTREC e ENDREC

STARTREC e ENDREC podem ser usados juntos ou separadamente.

Para cada cartão OUTFIL:
STARTREC=N começa o processamento no registro de número relativo N.
ENDREC=N termina o processamento no registro de número relativo N.

Para o 1° registro N=1 e N pode ir até 15 dígitos significativos.

O *default* para STARTREC é 1 e para ENDREC é o último registro do *dataset*.

Exemplos:

```
OUTFIL FNAMES=TOP10,ENDREC=10                Seleciona primeiros 10 registros
OUTFIL FNAMES=T500,ENDREC=500                Seleciona primeiros 500 registros
OUTFIL FNAMES=MEIO,STARTREC=501,ENDREC=1000  Seleciona registros de 501 a 1000
OUTFIL FNAMES=FIM,STARTREC=1001              Seleciona de 1001 até último reg.
```

11.4 INCLUDE - OMIT - SAVE

Os parâmetros INCLUDE e OMIT operam, para cada cartão OUTFIL, da forma usual, mas apenas para os registtos selecionados para a entrada (ver STARTREC e ENDREC).

SAVE recebe os registros não selecionados por nenhum cartão OUTFIL.

Exemplos:

```
OUTFIL FNAMES=J69,INCLUDE=(5,3,CH,EQ,C'J69')
OUTFIL FNAMES=J82,INCLUDE=(5,3,CH,EQ,C'J82')
OUTFIL FILES=01,OMIT=NONE
OUTFIL FNAMES=BI5NOT8,OMIT=(5,1,BI,EQ,B'00001000')
OUTFIL FNAMES=(OUT1,OUT2),
         OMIT=(7,2,CH,EQ,C'32',OR,18,3,CH,EQ,C'XYZ')
OUTFIL INCLUDE=(8,6,CH,EQ,C'CONTAB'),FNAMES=GP1
OUTFIL INCLUDE=(8,6,CH,EQ,C'CLIENT'),FNAMES=GP2
OUTFIL SAVE,FNAMES=NOTSELEC
```

11.5 OUTREC

O parâmetro OUTREC opera, para cada cartão OUTFIL, de uma forma muito mais poderosa do que o respectivo cartão, mas apenas para os registros selecionados para a entrada (ver STARTREC, ENDREC, INCLUDE e OMIT).

Para um uso corrente, pode ser usado da forma tradicional.

Exemplo:

```
OUTFIL FNAMES=OUT1,OUTREC=(5,21,30,50)
OUTFIL FILES=1,OUTREC=(5,20,31,40),
          ENDREC=2000,OMIT=(5,2,BI,EQ,18,2,BI)
```

11.6 CONVERT

O parâmetro CONVERT converte registros de tamanho variável em registros de tamanho fixo. Não esquecer que registros variáveis possuem 4 bytes de tamanho em seu início, portanto devemos criar os registros de saída a partir da posição 5 dos registros de entrada.

Exemplo:

```
OUTFIL FNAMES=OUTFIXO,CONVERT,OUTREC=(5,...)
```

11.7 SPLIT

O parâmetro SPLIT grava, alternadamente, um registro de saída entre todos os *datasets* de saída do cartão OUTFIL onde foi especificado.

Exemplo 1:

```
//SORT    EXEC PGM=SORT
//SYSOUT   DD  SYSOUT=*
//SORTIN   DD  DSN=BOOKS.INPUT,DISP=OLD
//SYSIN    DD  *
     OUTFIL FILES=(A,B),...
     OUTFIL FILES=(10,20,30),SPLIT, ...
/*
//SORTOFA     DD  DSN=BOOKS.OUTPUT1,DISP=(,CATLG)
//SORTOFB     DD  DSN=BOOKS.OUTPUT2,DISP=(,CATLG)
//SORTOF10    DD  DSN=BOOKS.OUTPUT10,DISP=(,CATLG)
//SORTOF20    DD  DSN=BOOKS.OUTPUT20,DISP=(,CATLG)
//SORTOF30    DD  DSN=BOOKS.OUTPUT30,DISP=(,CATLG)
```

SORTIN

11	20	34	56	78	99	100	200	250

SORTOF10

11	56	100

SORTOF20

20	78	200

SORTOF30

34	99	250

Exemplo 2:

```
//SORT    EXEC PGM=SORT
//SYSOUT   DD  SYSOUT=*
//SORTIN   DD  DSN=BOOKS.INPUT,DISP=OLD
//SYSIN    DD  *
     OUTFIL FNAMES=(DDN1,DDN2,DDN3,DDN4,DDN5,DDN6),SPLIT,  ...
/*
//DDN1     DD  DSN=BOOKS.OUTPUT.DDN1,DISP=(,CATLG)
//DDN2     DD  DSN=BOOKS.OUTPUT.DDN2,DISP=(,CATLG)
//DDN3     DD  DSN=BOOKS.OUTPUT.DDN3,DISP=(,CATLG)
//DDN4     DD  DSN=BOOKS.OUTPUT.DDN4,DISP=(,CATLG)
//DDN5     DD  DSN=BOOKS.OUTPUT.DDN5,DISP=(,CATLG)
//DDN6     DD  DSN=BOOKS.OUTPUT.DDN6,DISP=(,CATLG)
```

SORTIN

11	20	34	56	78	99	100	200	250

DDN1

11	100

DDN2

20	200

DDN3

34	250

DDN4

56

DDN5

78

DDN6

99

11.8 SPLITBY

O parâmetro SPLITBY grava, alternadamente, vários registros de saída entre todos os *datasets* de saída do cartão OUTFIL onde foi especificado.

Exemplo 1:

```
//SORT    EXEC PGM=SORT
//SYSOUT   DD  SYSOUT=*
//SORTIN   DD  DSN=BOOKS.INPUT,DISP=OLD
//SYSIN    DD  *
     OUTFIL FILES=(A,B),...
     OUTFIL FILES=(10,20,30),SPLITBY=3,   ...
/*
//SORTOFA      DD  DSN=BOOKS.OUTPUT1,DISP=(,CATLG)
//SORTOFB      DD  DSN=BOOKS.OUTPUT2,DISP=(,CATLG)
//SORTOF10     DD  DSN=BOOKS.OUTPUT10,DISP=(,CATLG)
//SORTOF20     DD  DSN=BOOKS.OUTPUT20,DISP=(,CATLG)
//SORTOF30     DD  DSN=BOOKS.OUTPUT30,DISP=(,CATLG)
```

SORTIN

11	20	34	56	78	99	100	200	250

SORTOF10

11	20	34

SORTOF20

56	78	99

SORTOF30

100	200	250

Capítulo 12

USO EFICIENTE DO DFSORT

Para obter-se melhor *performance* com o **DFSORT** seguir estas indicações:

- Planejar o desenvolvimento da aplicação (incluindo o formato dos dados) para o uso eficiente deste programa.
- Usar a melhor técnica para SORT/MERGE.
- Especificar as características dos arquivos de entrada e saída.
- Executar, sempre que possível, o **DFSORT** via JCL.
- Usar opções que possam melhorar a *performance* do **DFSORT**.
- Evitar opções que possam degradar a *performance* do **DFSORT**.
- Ser generoso com a memória virtual para o **DFSORT**.

Estas indicações serão discutidas a seguir.

APLICAÇÕES PLANEJADAS

Deve-se considerar vários fatores quando planejar uma nova aplicação:

BLOCAGEM EFICIENTE

A *performance* do **DFSORT** pode aumentar significativamente se os arquivos de entrada e saída forem blocados. Para grandes arquivos, arquivos em fita ou arquivos que são classificados freqüentemente, deve-se escolher uma blocagem alta. Em geral, quanto maior a blocagem para os arquivos de entrada e saída melhor será a *performance* do **DFSORT**.

MELHOR DESCRIÇÃO E DISPOSIÇÃO DOS CAMPOS

Quando uma nova aplicação for desenvolvida, a eficiência do **DFSORT** será melhor se:
1. definirem-se os campos no início do registro em ordem decrescente de significância.
2. descreverem-se os campos e formato dos dados o mais eficientemente possível.

O exemplo a seguir ilustra o benefício da localização do campo no início do registro.

Assumir que o registro de entrada tenha o seguinte *layout*:

A	1	B	2	C

onde : 1 = o campo mais significante para SORT ou MERGE
2 = o campo menos significante para SORT ou MERGE

A, B e C não são campos para SORT ou MERGE. Internamente, é util que o programa reorganize os campos antes do SORT ou MERGE da seguinte forma:

1	2	A	B	C

Após SORT ou MERGE os campos podem ser restaurados para a posição original. Desta forma, com o segundo diagrama irá aumentar a eficiência do seu programa.

Para o formato dos campos e descrição, sempre que possível:

- usar campos em caracter EBCDIC ou binário;
- colocar campos binários em limite de byte;
- evitar alterar a ordem natural de classificação, pois esta função não somente aumenta o uso de CPU, mas também aumenta o tamanho interno do registro;
- usar o formato decimal compactado ao invés de decimal zonado, porque o **DFSORT** compacta campos e, com campos em decimal zonado, tem-se um aumento no tamanho interno do registro;
- se vários campos são contíguos, estão na ordem correta para a classificação e serão classificados na mesma ordem (ascendente ou descendente), especificá-los como um único campo;
- evitar a superposição de campos para classificação.

USAR A MELHOR TÉCNICA PARA SORT/MERGE

Dependendo de várias condições, o **DFSORT** seleciona diferentes técnicas para classificar ou fundir. A mensagem ICE143I informa qual técnica foi escolhida.

. Técnicas de Classificação

Uma condição que afeta a escolha da técnica de classificação é o tipo de área de trabalho utilizada: disco ou fita (pouco utilizado). As técnicas *blockset*, *peerage* e *vale* só utilizam área de trabalho em disco. Se se usar área de trabalho em fita, o **DFSORT** escolherá a técnica menos eficiente de classificação.

. Técnica blockset do **DFSORT**

Para registros de tamanho fixo o **DFSORT** usar a técnica *flr-blockset* que é a mais eficiente para este tipo de registro. Se uma ou mais condições não for satisfeita para a técnica *flr-blockset* (por exemplo, se o campo for muito longo), a técnica *peerage* ou *vale* é usada. Agora, para registros de tamanho váriavel o **DFSORT** usar a técnica *vlr-blockset* que dá a melhor *performance* no manuseio desses registros. Se uma ou mais condições não for satisfeita (por exemplo, se o campo for muito longo) a técnica *vale* é usada.

. Registros fixos ou váriaveis.

Obs : se a técnica *blockset* não for selecionada usar o cartão
//SORTDIAG DD SYSOUT=*, para saber o motivo da não escolha.

. Técnica convencional para Merge

Se as condições para a técnica *blockset* não for satisfeita (por ex. campo longo) o **DFSORT** usa a técnica convencional.

ESPECIFICAR AS CARACTERÍSTICAS DOS ARQUIVOS DE ENTRADA E SAÍDA

O **DFSORT** procura ser o mais eficiente possível com base nas informações que lhe são oferecidas. Se essas informações não forem corretas, o programa assume valores que podem torná-lo significativamente ineficiente. Para isto observar as seguintes informações úteis:

. Simplificar a descrição dos campos de controle

Quando projetar o formato dos registros, planejá-los de tal forma que a classificação ou fusão seja eficiente. Por exemplo, sempre que possível, construir campos em EBCDIC ou em binário (começando e terminando em limite de byte): isto diminue o tempo de processamento. Dados em ponto fixo, compactado ou decimal zonado podem ser classificados como se fossem binários se eles forem sempre positivos; e, para dois ou mais campos contíguos classificados do mesmo jeito(**A** / **D**), classificá-los como um só.

. Tamanho do arquivo

Quando o **DFSORT** obtem informações a respeito do tamanho dos arquivos, ele pode fazer o uso mais eficiente das áreas de trabalho e memória virtual. Esta informação também é importante quando as áreas de trabalho são alocadas dinamicamente.

Se se conhece o número exato de registros para o **DFSORT**, usá-lo para o parâmetro FILSZ no cartão de controle OPTION ou SORT. Se não se conhece esse número exatamente, fornecer um valor o mais próximo possível.

. Registros de tamanho váriavel

Para registros de tamanho variável é interessante observar se o parâmetro LRECL do DCB corresponde ao tamanho máximo real do registro do seu arquivo.

EXECUTAR O DFSORT VIA JCL (SEMPRE QUE POSSÍVEL)

Como uma regra, o **DFSORT** é mais eficiente quando executado via JCL do que chamado via programa (*sort* interno).

Embora chamar o **DFSORT** via programa possa ser conveniente, principalmente se ele modifica os dados antes e depois da classificação, deve-se saber que esta não é uma forma eficiente para a utilização do **DFSORT**.

Chamando o **DFSORT** via JCL consegue-se fazer algumas modificações nos dados, antes e depois do SORT, se se usar os cartões INCLUDE/OMIT, INREC/OUTREC e SUM.

USAR OPÇÕES QUE POSSAM MELHORAR A PERFORMANCE DO DFSORT

. Cartões INCLUDE ou OMIT - opções STOPAFT e SKIPREC

Pode-se usar os cartões INCLUDE ou OMIT e a opção STOPAFT ou SKIPREC para reduzir o tamanho do arquivo de entrada, o que reduz o tempo para a transferência dos dados.

- . Os cartões INCLUDE e OMIT permitem selecionar registros pela comparação de campos com constantes e/ou outros campos.

- . A opção STOPAFT permite específicar o número maximo de registros aceitos para a classificação ou cópia.

- . A opção SKIPREC permite desprezar alguns registros do início do arquivo que será classificado ou copiado.

. Cartões INREC e OUTREC

Pode-se usar o cartão INREC para reformatar os registros de entrada antes do SORT. Com isto pode-se tornar o processamento mais eficiente, se se reduzir o tamanho dos registros (ou menos eficiente, se se aumentar o tamanho dos registros).

Com o cartão OUTREC pode-se aumentar o tamanho dos registros depois do SORT, alinhando os campos e introduzindo brancos para separar campos e tornar a saída mais legível.

Quando os cartões INREC e/ou OUTREC são usados com os cartões INCLUDE ou OMIT, somente os registros que passarem pelo criterio de seleção (INCLUDE ou OMIT) serão reformatados.

Existem tres tipos de campos que podem ser removidos pelos cartões INREC/OUTREC:

1. campos com brancos ou zeros binários podem ser removidos antes do SORT pelo uso do cartão INREC, e em seguida podem ser reinseridos após o SORT pelo uso do cartão OUTREC;
2. campos que não são necessários para os registros de saída podem ser removidos antes do SORT;
3. se o arquivo contém registros variáveis e somente a parte fixa é necessária para a sua aplicação, então a parte variável pode ser eliminada antes do SORT. Isto permite que o **DFSORT** manuseie registros com tamanho fixo e, como resultado, teremos uma melhor eficiência.

. Cartão SUM

Pode-se usar o cartão SUM para somar valores em alguns campos do registro. Com isso teremos: uma redução no tamanho do arquivo para ser classificado ou fundido, e uma redução no tempo de processamento e transferência de dados.

EVITAR OPÇÕES QUE POSSAM DEGRADAR A PERFORMANCE DO DFSORT

Tais opções são :

VERIFY - a opção VERIFY afeta a performance negativamente, porque ela envolve uma operação extra de leitura do arquivo de saída.

EQUALS - a opção EQUALS adiciona quatro bytes de controle para cada registro do arquivo de entrada, com isto teremos um aumento no tempo de processamento e transferência dos dados. Isto não se aplica para a técnica *blockset* para classificar registros de tamanho váriavel que usa opção EQUALS.

NOWRKSEC - a opção NOWRKSEC não permite extenção automática das áreas de trabalho. Isto irá causar reutilização dos *extents* envolvidos com operações extras de leitura e/ou gravação.

NOBLKSET - a opção NOBLKSET impede a utilização da técnica mais eficiente do **DFSORT**, que é a *blockset*.

CKPT - a opção CKPT impede a utilização da técnica *blockset* que é a mais eficiente do **DFSORT**.

SER GENEROSO COM A MEMÓRIA VIRTUAL PARA O DFSORT

Em geral, quanto mais memória virtual para o **DFSORT** (em torno de 1 megabyte) melhor é a sua performance.

Pode-se alocar memória virtual usando o cartão de controle OPTION e específicando o parâmetro MAINSIZE=nK onde n é o total de memória virtual para ser alocada pelo **DFSORT**. Ex: OPTION MAINSIZE=800K

Outra forma de alocar memória virtual é usando parâmetro PARM do JCL e específicando SIZE=nK. Ex:

```
//SORT1  EXEC PGM=SORT,PARM='SIZE=500K'
```

Se a quantidade de registros para serem classificados for razoavelmente pequena em comparação com a quantidade de memória virtual fornecida, o **DFSORT** não usará área de trabalho, pois a classificação será feita em memória. Mas nem sempre é possível classificar um arquivo sem usar áreas de trabalho. Sendo assim, para o **DFSORT** existem duas maneiras de alocar áreas de trabalho:

1. ALOCAÇÃO DE ÁREA DE TRABALHO VIA JCL

Uma alocação adequada de área de trabalho para o **DFSORT** é a seguinte: em geral, alocar a área de trabalho com o dobro do espaço usado pelo arquivo de entrada. Caso essa área não for suficiente, e não se fornecer alocação secundária, o **DFSORT** tem como *default* de alocação secundária 25% da área primária.

---

2. ALOCAÇÃO DE ÁREA DE TRABALHO DINAMICAMENTE

A alocação dinâmica da área de trabalho pode ser solicitado via cartão de controle do **DFSORT** ou via cartão OPTION.

VIA CARTÃO DE CONTROLE

```
          SORT FIELDS=(........),DYNALLOC=(d,n),SIZE=y

      ou  SORT FIELDS=(........),DYNALLOC=(d,n),SIZE=Ex
```

onde: y = número exato de registros
x = número aproximado de registros
d = tipo de disco, usualmente SYSDA
n = número de áreas de trabalho (n entre 1 e 16).

VIA CARTÃO OPTION

```
           OPTION DYNALLOC=(d,n),SIZE=y

      ou   OPTION DYNALLOC=(d,n),SIZE=Ex
```

onde: y = número exato de registros
x = número aproximado de registros
d = tipo de disco, usualmente SYSDA
n = número de áreas de trabalho (n entre 1 e 16).

Caso se erre na quantidade de registros ou na quantidade aproximada de registros e não seja fornecida alocação secundária, o **DFSORT** aloca 25% da área primária. Mas na proxima execução do **DFSORT** convém corrigir esse erro, caso contrário o **DFSORT** estará sendo usado ineficientemente.

Capítulo 13

UTILITÁRIO ICETOOL

O utilitário ICETOOL, disponível no **DFSORT**, tem a capacidade de explorar várias funções deste produto, onde o usuário pode executar **operações múltiplas**, em **um ou mais arquivos**, num **único *step***.

OPERADORES DO ICETOOL

Os operadores do ICETOOL são em número de 12 e, com eles, criam-se aplicações que executam várias tarefas complexas usando o **DFSORT**.

Os operadores que mais nos interesssam são: SORT e COPY; os outros 10 são operadores estatísticos (Ex: COUNT e STATS).

SORT: classifica um arquivo para mais de um arquivo de saída.

COPY: copia um arquivo de entrada para mais de um arquivo de saída.

COUNT: imprime quantidade de registros de um arquivo.

STATS: imprime o mínimo, máximo, médio e o total de qualquer campo numérico de um arquivo.

ARQUIVO DE ENTRADA

Os operadores acima requerem um arquivo de entrada. O arquivo usado por um operador pode ser o mesmo ou diferente do arquivo de entrada usado por outro operador. Desta forma, **ICETOOL pode processar vários arquivos num único *step***.

JCL REQUERIDO PARA O ICETOOL

Para executar-se o utilitário ICETOOL é necessário codificar o seguinte JCL:

```
//EXAMP  JOB  ......
//TOOL   EXEC PGM=ICETOOL
//TOOLMSG DD SYSOUT=*
//DFSMSG  DD SYSOUT=*
//TOOLIN  DD  *
      < COMANDOS DO ICETOOL >
/*
   < CARTÕES DE JCL ADICIONAIS >
```

Onde:

- TOOLMSG é o DDNAME do arquivo de saída para as mensagens do pgm ICETOOL.
- DFSMSG é o DDNAME do arquivo de saída para as mensagens do pgm **DFSORT**
- TOOLIN é o DDNAME do arquivo de entrada que deve conter os operadores do ICETOOL e, opcionalmente, comentários e/ou cartões em brancos.
- os DDNAMES adícionais são os arquivos que serão utilizados pelo ICETOOL.

CARTÕES EM BRANCO E COMENTÁRIOS PARA O ICETOOL

- Cartões em branco e comentários podem ser colocados em qualquer lugar entre os cartões de operação do ICETOOL.
- Os cartões de comentário começam com * na coluna 1 e serão impressos com os demais cartões do ICETOOL em TOOLMSG.
- Os cartões em branco das colunas 1 até 72, serão ignorados e impressos em TOOLMSG com os demais cartões de operação do ICETOOL.

Para escrever um cartão em branco e um de comentário:
1) Pular uma linha após o cartão TOOLIN;
2) Por * na coluna 1, seguido de um comentário.

```
//TOOLIN  DD *

*   ESTATÍSTICAS DE TODAS FILIAIS
/*
```

APRENDENDO A UTILIZAR O PROGRAMA ICETOOL

Considerar o arquivo "*SORT*.BRANCH" para efeito de exemplo e explicação de alguns operadores do program ICETOOL.

A figura 1 mostra o tamanho e formato de cada campo do arquivo das filiais ("*SORT*.BRANCH").

FIGURA 1 - TAMANHOS E FORMATOS DOS CAMPOS DO ARQUIVO DE FILIAIS

CAMPO	TAMANHO	FORMATO DOS CAMPOS
CIDADE	15	CH
ESTADO	2	CH
FUNCIONÁRIOS	4	ZD
RENDA	6	PD
LUCRO	6	PD

A figura 2 mostra os registros do arquivo "*SORT*.BRANCH".

FIGURA 2 - REGISTROS DO ARQUIVO "*SORT*.BRANCH"

CIDADE	ESTADO	FUNCIONÁRIOS	RENDA	LUCRO
1 15	16 17	18 21	22 27	28 33
LOS ANGELES	CA	32	22530	-4278
SAN FRANCISCO	CA	35	42820	6832
FORT COLLINS	CO	22	12300	-2863
SACRAMENTO	CA	29	42726	8276
SUNNYVALE	CA	18	16152	-978
DENVER	CO	33	31876	6288
BOULDER	CO	32	33866	7351
MORGAN HILL	CA	15	18200	3271
VAIL	CO	19	23202	5027
SAN JOSE	CA	21	27225	8264
SAN DIEGO	CA	22	32940	8275
ASPEN	CO	20	25800	5200

CRIANDO MÚLTIPLOS ARQUIVOS CLASSIFICADOS

Pode-se usar o operador SORT do ICETOOL para criar arquivos classificados. Um único operador SORT pode criar até 10 arquivos de saída idênticos. Usando comandos de INCLUDE ou OMIT do **DFSORT**, é possível selecionar um sub-conjunto de registros do arquivo de entrada. Bem como usando comandos INREC ou OUTREC, é possível rearranjar os campos de um arquivo.

Para escrever operadores SORT que criem dois arquivos; um em disco e outro em fita, para as filiais da Califórnia e do Colorado, classificando por cidade temos:

. Estado Califórnia :

```
                    SORT FROM(SEQENT1 ) TO(CADASD,CATAPE) USING(CACS)
                                 |            |       |             |
   DDNAME do arquivo de entrada <--            |       |             |
   DDNAME do primeiro arquivo de saída <-------        |             |
   DDNAME do segundo arquivo de saída <----------------              |
   prefixo CACS do DDNAME CACSCNTL para o arquivo <-----------------
            que contém os comandos do DFSORT
```

. Estado Colorado :

```
                    SORT FROM(SEQENT1 ) TO(CODASD,COTAPE) USING(COCS)
                                 |            |       |             |
   DDNAME do arquivo de entrada <--            |       |             |
   DDNAME do primeiro arquivo de saída <-----          |             |
   DDNAME do segundo arquivo de saída <----------------              |
   prefixo COCS do DDNAME COCSCNTL para o arquivo <-----------------
            que contém os comandos do DFSORT
```

. Comandos do **DFSORT** para Califórnia fica:

```
   SORT FIELDS=(1,15,CH,A)
   INCLUDE COND=(16,2,CH,EQ,C'CA')
```

. Comandos do **DFSORT** para Colorado fica:

```
   SORT FIELDS=(1,15,CH,A)
   INCLUDE COND=(16,2,CH,EQ,C'CO')
```

O JCL completo fica da seguinte forma:

```
//EXAMPLE JOB (.......
//TOOL    EXEC  PGM=ICETOOL
//SEQENT1 DD DSN=SORT.BRANCH,DISP=SHR
//TOOLMSG  DD  SYSOUT=*
//DFSMSG   DD  SYSOUT=*
//TOOLIN   DD  *

*-----------------------------------
*   CRIA ARQUIVOS PARA CALIFORNIA
*-----------------------------------
  SORT FROM(SEQENT1) TO(CADASD,CATAPE) USING(CACS)

*-----------------------------------
*   CRIA ARQUIVOS PARA COLORADO
*-----------------------------------
  SORT FROM(SEQENT1) TO(CODASD,COTAPE) USING(COCS)
/*
//CACSCNTL DD  *
   SORT FIELDS=(1,15,CH,A)
   INCLUDE COND=(16,2,CH,EQ,C'CA')
/*
//CADASD   DD  DSN=D.CA.BRANCH,DISP=(,CATLG),SPACE=(TRK,2),UNIT=SYSDA
//CATAPE   DD  DSN=F.CA.BRANCH,DISP=(,CATLG),UNIT=TAPECART
//*
//COCSCNTL DD  *
   SORT FIELDS=(1,15,CH,A)
   INCLUDE COND=(16,2,CH,EQ,C'CO')
/*
//CODASD   DD  DSN=D.CO.BRANCH,DISP=(,CATLG),SPACE=(TRK,3),UNIT=SYSDA
//COTAPE   DD  DSN=F.CO.BRANCH,DISP=(,CATLG),UNIT=TAPECART
//*
//
```

CRIANDO MÚLTIPLAS CÓPIAS DE UM ARQUIVO

Para se criar múltiplas cópias de um arquivo, usar o operador COPY do programa ICETOOL. O operador COPY não requer qualquer comando **DFSORT**, mas pode-se utilizá-los, caso necessite (ex: OMIT, INREC, INCLUDE, OUTREC).

Dois casos do operador COPY com o respectivo JCL:

```
//SEQENT1 DD DSN=SORT.BRANCH,DISP=SHR
//TOOLIN  DD  *
        COPY FROM (SEQENT1) TO(D1,D2,D3)
        COPY FROM (SEQENT1) TO(P1) USING (COPY)
/*
//D1       DD  DSN=SORT.BRANCH.COPY1,DISP=(,CATLG),SPACE=(TRK,2),UNIT=SYSDA
//D2       DD  DSN=SORT.BRANCH.COPY2,DISP=(,CATLG),SPACE=(TRK,2),UNIT=SYSDA
//D3       DD  DSN=SORT.BRANCH.COPY3,DISP=(,PASS),SPACE=(TRK,2),UNIT=SYSDA
//P1       DD  SYSOUT=*
//COPYCNTL DD  *
       INCLUDE COND=(16,2,CH,EQ,C'CA')
/*
```

No exemplo acima temos:
  . O primeiro operador COPY cria tres cópias do arquivo "SORT.BRANCH".
  . O segundo operador COPY imprime os registros que contém as filiais do estado da Califórnia do arquivo "*SORT*.BRANCH". Notar que somente os campos em caracter serão legíveis.

QUANTIDADE DE REGISTROS DE UM ARQUIVO

Pode-se imprimir a quantidade de registros de um arquivo com o uso do comando COUNT.
Para determinar a quantidade de registros do arquivo "SORT.BRANCH" escreva:

```
                      COUNT FROM(SEQENT1)
                                    |
 DDNAME DO ARQUIVO DE ENTRADA <------
```

Para saber quantas filiais estão no estado da Califórnia, escrever:

```
                       COUNT FROM(SEQENT1) USING(CAIN)
                                    |             |
     DDNAME DO ARQUIVO DE ENTRADA  <--             |
     COMANDOS DO DFSORT <--------------------------
```

Para o JCL do **DFSORT** teremos :

```
//CAINCNTL  DD  *
      INCLUDE COND=(16,2,CH,EQ,C'CA')
/*
```

IMPRIMINDO ESTATÍSTICAS DE CAMPOS NUMÉRICOS

Com o uso do operador STATS pode-se obter informações estatísticas de no máximo dez campos numéricos.

Para escrever um comando STATS e imprimir estatísticas dos campos "funcionários", "lucro" e "renda" do arquivo de filiais:

```
                         STATS FROM(SEQENT1) ON(18,4,ZD) ON(28,6,PD) ON(22,6,PD)
                                      |      |           |           |
DDNAME DO ARQUIVO DE ENTRADA <--------      |           |           |
FUNCIONÁRIOS <-------------------------------           |           |
LUCRO <-------------------------------------------------           |
RENDA <-----------------------------------------------------------
```

STATS imprime, em TOOLMSG, os valores: máximo, minimo, médio e total, de cada campo numérico referenciado.

O JCL para o exemplo acima seria:

```
//EXAMP    JOB (.......
//TOOL    EXEC  PGM=ICETOOL
//TOOLMSG DD  SYSOUT=*
//DFSMSG  DD  SYSOUT=*
//TOOLIN  DD  *
*   ESTATÍSTICAS DE TODAS FILIAIS
  STATS FROM(SEQENT1) ON(18,4,ZD) ON(28,6,PD) ON(22,6,PD)
/*
//SEQENT1 DD DSN=SORT.BRANCH,DISP=OLD
```

CONTINUAÇÃO DOS COMANDOS

Se o operador STATS (ou qualquer outro operador do ICETOOL) não couber entre as colunas 1 e 72, pode-se continuá-lo em múltiplas linhas. Basta colocar, após cada operador ou operando, um hifen (-) e a próxima linha é tratada como continuação. Qualquer caracter especificado após o hifen será ignorado, o que permite comentários.

Note que o operador ou cada operando deve ser completamente específicado em uma única linha (entre as colunas 1 e 72).
Ex:

```
       STATS             -   este é o operador
          FROM(SEQENT1)  -   SEQENT1 é o DDNAME para "SORT.BRANCH"
             ON(18,4,ZD) -
             ON(28,6,PD) -
             ON(22,6,PD)
```

Capítulo 14

USANDO SÍMBOLOS

Um símbolo é um nome, de preferência significativo e claro, que pode ser usado para representar um campo ou uma constante. Conjuntos de símbolos podem ser usados para descrever um grupo de campos e constantes associados a um ou mais tipos de registro.

CRIANDO O SYMNAMES DATASET

DFSORT e ICETOOL obtêm os símbolos a serem usados do *dataset* especificado em um cartão SYSMNAMES DD.

O *dataset* SYSMNAMES deve ser criado com RECFM=FB e LRECL=80; pode ser usado, também, um cartão SYSMNAMES DD *.

O *dataset* SYMNOUT é opcional e especifica um dataset onde o **DFSORT** listará o conteúdo original de SYMNAMES e a tabela de símbolos construída a partir dele.

DEFININDO SÍMBOLOS

Exemplo 1:

```
* CAMPOS
NOME_COMPLETO,6,40,CH
  NOME,=,20,CH
  SOBRENOME,*,=,=
CONTA_CORRENTE,53,3,PD
SKIP,2
BALANCO,*,6,ZD
TIPO,*,12,CH

* CONSTANTES
EMPRESTIMO,'EMPRESTIMO'
VERIFICACAO,'VERIFICACAO'
NIVEL1,50000
NIVEL2,-100
```

Linhas em branco são permitidas.

* na 1ª. coluna, indica um comentário;
em outra coluna, indica a posição seguinte ao último byte utilizado.

= indica a posição, o tamanho ou o formato utilizado na linha anterior; caso seja usado para indicar a mesma posição da linha anterior, permite redefinição de campos.

SKIP,n indica um *filler* de n posições (n pode ir de 1-32752); útil quando se utiliza * na linha seguinte à do SKIP.

Exemplo 2:

```
POSITION,27
BALANCO,*,5,PD
BALANCO_ID,*,8,CH
POSITION,84
NOVO_BALANCO,=,20,CH
```

`POSITION,q` posiciona a próxima posição a ser referenciada.

`SYMNOUT` mostrará a seguinte tabela de símbolos:

```
BALANCO,27,5,PD
BALANCO_ID,32,8,CH
NOVO_BALANCO,84,20,CH
```

Exemplo 3:

```
WK_AREA,20,100
  ID,=,10,CH
  NOME,20,CH
POSITION,WK_AREA
  STATUS,=,1,CH
  ENDERECO,*,60,CH
```

`POSITION,símbolo` posiciona a próxima posição a ser refenciada no início de "símbolo".

`SYMNOUT` mostrará a seguinte tabela de símbolos:

```
WK_AREA,20,100
ID,20,10,CH
NOME,30,20,CH
STATUS,20,1,CH
ENDERECO,21,60,CH
```

USANDO SÍMBOLOS

```
//SORT01  EXEC PGM=SORT
//SYSOUT DD SYSOUT=A
//SYMNAMES DD *
* CAMPOS
NOME_COMPLETO,6,40,CH
  NOME,=,20,CH
  SOBRENOME,*,=,=
CONTA_CORRENTE,53,3,PD
SKIP,2
BALANCO,*,6,ZD
TIPO,*,12,CH

* CONSTANTES
EMPRESTIMO,'EMPRESTIMO'
VERIFICACAO,'VERIFICACAO'
NIVEL1,50000
NIVEL2,-100
/*
//SYMNOUT DD SYSOUT=*
//SORTIN DD DSN=------,DISP=SHR
//SORTOUT DD DSN=-----,DISP=SHR
//SYSIN DD *
  INCLUDE COND=((TIPO,EQ,EMPRESTIMO,AND,BALANCO,GT,NIVEL1)OR,
                TIPO,EQ,VERIFICACAO,AND,BALANCO,LE,NIVEL2))
  SORT FIELDS=(SOBRENOME,A,NOME,A,
               TIPO,A,CONTA_CORRENTE,D)
/*
```

SÍMBOLOS

Formato: símbolo,valor [comentário]

Regras de codificação:
1. Podem ser utilizadas as colunas de 1 a 80.
2. O símbolo pode começar em qualquer coluna.
3. O comentário é opcional, mas deve ser separados do valor por, ao menos, um espaço; deve caber na mesma linha ou deve-se utilizar uma linha de comentário (* na posição 1).
4. Cada símbolo e seu valor devem ser codificados em uma só linha (continuação não é permitida).
5. O valor pode ser uma constante ou um campo.
6. O símbolo pode ter de 1 a 50 caracteres; são permitidos: maiúsculas (A-Z), minúsculas (a-z), números (0-9) e certos caracteres especiais (#,$,@,_,-); o primeiro caracter não pode ser numérico ou hífen.
   Símbolos são *case-sensitives*.

CONSTANTES

Uma constante pode ser do tipo caracter, hexadecimal, *bit string* ou número decimal.
Para maiores detalhes ver página 12.

CAMPOS

Um campo pode, e deve, ser definido com posição, tamanho e formato e o **DFSORT** substituirá apenas a posição e tamanho onde for necessário.