Using DFSORT and ICETOOL

Using DFSORT and ICETOOL - Course Objectives

On successful completion of this class, the student, with the aid of the appropriate reference materials, should be able to:

1. Write JCL and DFSORT control statements to sort, copy, and merge records in the following kinds of files as input, and to produce these kinds of files as output:

    * Sequential data sets
    * VSAM data sets
    * Members of PDS or PDSE
    * Files in the z/OS File System (zFS - UNIX file support)

2. Create symbolic name files to use in DFSORT and ICETOOL control statements

3. Write JCL and DFSORT and ICETOOL control statements that enable you to perform these tasks with no need to write code in a programming language:

    * Work with subsets of files (filter / extract out records)
    * Build new records, reformat existing records, even recognizing and handling different record layouts in the same file; including:
        - adding, removing, and reordering fields
        - converting data type of fields, and editing content
        - doing arithmetic calculations, inserting sequence numbers
        - replace values using lookup pairs in specific locations
        - replace values in any location
    * Extract fixed length values from variable length fields (parse)
    * Output multiple files in a single pass, including XML, HTML, and reports with up to three levels of control breaks
    * Working with dates and times in a wide variety of formats, including SMF, TOD, and ETOD, and working with dates with two digit years
    * Convert betweeen fixed length record files and variable length record files
    * Join records from two files and process the resulting record set
    * Use locales for sorting and copying, and work with ASCII files
    * Combine fields from two or more files into records in a single file.

Using DFSORT and ICETOOL - Topical Outline

Day One

Day Three

Note: this course is a thorough introduction to the facilities of DFSORT and ICETOOL, omitting discussions of exits and the Extended Function Support.

While we cover all the control statements and operands in varying degree, with lots of examples, you should consider this a starting point.

The next stop on your journey would be to examine the publications on the DFSORT website, especially the "DFSORT Application Programming Guide", which is the definitive source for this material, and "Smart DFSORT Tricks", which provides additional examples of applications for DFSORT and ICETOOL based on requests from customers.

You can find these publications, and more, on the DFSORT website beginning at:

https://www.ibm.com/support/pages/dfsort then follow the links to details.
for the Smart Tricks: https://www.ibm.com/support/pages/node/665475

Section Preview

- **Introduction to DFSORT**

  Background

  Setting up for labs (Machine Exercise)

SORT - Background

❒ **When OS/360 first came out (this was the ancestor of today's z/OS), it included a free sort/merge utility as part of the operating system**

But it wasn't very good:

✗ Competitors came out and were charging for their sort products and were succeeding, even against the free sort included with OS/360!

❒ **At some point, IBM either got embarrassed or they saw a potential revenue stream and they started paying attention to their sort**

❒ **This competition was good, since all the major players competed against each other, each improving performance and functionality**

Often taking turns leapfrogging each other as the "best" available sort product

Today, the major players in the Sort market are IBM's DFSORT and SyncSort from Syncsort, Inc.

✗ At least one other sort package is in the fray, but these two have, by far, the largest market share

❒ **Another nice feature of the competition: often these two products have the same syntax for control statements: this makes it easier for a customer to switch from one to the other!**

❒ **So, while this course is about DFSORT, much of the content also applies to SyncSort**

SORT - Background, 2

❒ **Today, DFSORT has evolved to be a very sophisticated tool that can:**

<u>Sort</u> the records in a file into a desired sequence based on one or more <u>sort keys</u>

- ✗ The records can arrive from a sequential file, a VSAM file, a member of a PDS or PDSE, or a file in the zFS (z/OS File System - files used by z/OS UNIX System Services)
  - ➢ Or they can be inserted from any other source using a locally written exit routine
- ✗ You can sort all the records, or just some of the records
- ✗ You can reformat the input records before or after sorting
  - ➢ You can process records after sorting but before the actual output writing using a locally written exit routine
- ✗ The records can go to multiple output files
- ✗ The output file(s) can be formatted using a rich collection of field editing, page formatting, and report building features

<u>Merge</u> records from multiple files into a single file - as long as all the records have the same general layout and are in the same relative sort sequence by one or more collation keys

<u>Copy</u> the records from one file to another

❒ **Sort, Copy and Merge can all filter records as described above, and convert between FB and VB records**

❒ **DFSORT comes with a powerful utility, ICETOOL, and a high speed replacement for IEBGENER called ICEGENER**

These are discussed in this course also

Computer Exercise: Setting up for labs

This lab establishes the environment for running all the subsequent labs in this course. To do this, you will run a little dialog that:

1. Establishes the naming conventions to use for your data sets
   (default: <*tsoid*>**.TR.***name*)

2. Establish the zFS directories to use for holding zFS files for input and output for sort operations
   (default: **/<u/**<*tsoid*>**/SortData)**

   (Note: this is bypassed if you do not have access to the zFS, or if you blank out the directory name in the dialog when you run it)

The dialog will then create the following files for you:

<hlq>.TR.CONTACTS	(flat file test data)
<hlq>.TR.CNTL	(JCL library with some members already there)
<hlq>.TR.CUSTOMER	(flat file test data)
<hlq>.TR.DATA	(data library with some members already there)
<hlq>.TR.INPUTA	(flat file test data, character, packed, binary data)
<hlq>.TR.INPUTX	(flat file test data, character, packed, binary data)
<hlq>.TR.INPUTE	(flat file test data, variable length records)
<hlq>.TR.LOCS	(flat file test data)
<hlq>.TR.ORDERS	(flat file test data)
<hlq>.TR.ZINPUTA	(flat file test data, character data only)
<hlq>.TR.ZINPUTX	(flat file test data, character data only)
<hlq>.TR.ZASCIIX	(flat file test data, ASCII version of above file)

If you will be running the zFS-related labs, the dialog will also create the two directories mentioned above.

To run this dialog, from ISPF option 6 enter this command:

```
===> exec '_____________.train.library(b625strt)' exec
```

Note that the record layouts for all files are in the Appendix to this book.