



CCAPRINT

A Newsletter Excerpt for System 1032 Users

November 2006

USE OF AND ACCESS TO PRODUCTS AND FEATURES ARE IN ACCORDANCE WITH THE TERMS AND CONDITIONS OF THE USER'S SOFTWARE LICENSE. THE PRESENTATION OF MATERIAL HEREIN DOES NOT, IN ANY MANNER, MODIFY SUCH TERMS AND CONDITIONS.

Using FIND DUPLICATES for More Than One Attribute

By Tym Stegner

The FIND DUPLICATE command is useful for locating duplicate records for a particular keyed attribute. Its limitation is that it can operate upon only a single, keyed attribute. This article discusses and demonstrates a method to locate multiple duplicate attributes.

A user recently requested any information available about how to get the FIND DUPLICATES command to work on more than one attribute at a time. Her problem being that she needed to find duplicate records within an addresses dataset, matching upon first name, last name, and middle name. She had used a FIND DUPLICATES command to locate duplicate last-names, but she needed the other two attributes to form the unique combinations. Furthermore, the middle-name attribute was not keyed.

Considering the Nesting Consider Sets Solution

At first it appeared that the means to address this situation involved nesting consider sets combined with the use of the FIND DUPLICATES command. This was not the case, as Figure A illustrates. In this example, lets look at the FN/MN/LN scenario described by our customer.

Figure A. The FN/MN/LN scenario

```
$ S1032 Open Ds MONAMES
Current dataset is now MONAMES
Computer Corporation of America System 1032 Version V9.81-1
Copyright 2002, Computer Corporation of America
1032> Find Dup Last_Name
604 MONAMES records found
1032> Consider On
1032> Find Dup First_Name
471 MONAMES records found
1032>
```

Starting with the FIND DUP Last_Name command, the initial selection is only those records having two or more matching last names. A consider set is established to restrict subsequent query to those records, then the next command - FIND DUP First_Name - is performed.

This appears to have narrowed our selection. However, as there is no logical connection between the first and last names, this supposition is in error. This only identifies duplicate first names within our previously selected records, but it does not imply these duplicates in any way belong to the last names already selected.

The FIND DUP, plus DROP \$ID, Solution

While the initial FIND DUP Last_Name command, as shown in Figure A, can be a starting point, we need a comparison algorithm that will do programmatically what it is very easy to do visually. This algorithm starts with a SORT on the attributes that make up the selection criteria: FN, MN, LN. Figure B illustrates if we were to print such a selection of records.

Figure B. Printing the sorted records

```

1032> Find Dup LN
604 MONAMES records found
1032> Sort LN FN MN
1032> Print LN FN MN
      Last Name           First Name           Middle Name
-----
Aagad                Lene                ??????????????????
Aagad                Lene                ??????????????????
Adams                Ken                 ??????????????????
Adams                Shari              ??????????????????
Administrator       Your System        ??????????????????
Administrator       Your System        ??????????????????
Administrator       Your System        ??????????????????
Allen                Brad               ??????????????????
Allen                Jim                ??????????????????
Allen                Keith              ??????????????????
Ambrus               Mark               ??????????????????
Ambrus               Mark               ??????????????????
Anderson            Bob                ??????????????????
Anderson            Jane               ??????????????????
Anderson            Tessa             ??????????????????
Anderson            Wayne              ??????????????????
Bailey              Denise             ??????????????????
Bailey              Howard             ??????????????????
Bailey              Vera               ??????????????????
Baker               Homer              ??????????????????
Baker               Stephen            ??????????????????
. . .
1032>

```

The way our eyes pick out duplicate records is to note changes in the data values. Our algorithm must perform a similar task. The following code is the MULTIDUP.DMC file which can accomplish this.

The MULTIDUP.DMC file

The basic idea is to store in variables a potentially matching set of data values, plus their locator \$ID, then progress sequentially through the selection set, and make note of when the value comparisons do not match.

```

BEGIN

VARIABLE xFN,xMN TEXT 20      !declare to match candidate attrs
VARIABLE xLN      TEXT 30

VARIABLE lastWasDup,noMoreRecs      LOGICAL INITIALLY FALSE
VARIABLE spc                        TEXT 1  INITIALLY " "
VARIABLE xId,dupsCount              INTEGER INITIALLY 0

FIND DUPLICATES Last_Name          !primary is OK
SORT Last_Name First_Name Middle_Name ! collect by groups

GETRECORD                          !set initial match record

LET xLN = $CVTMISS(Last_Name,spc),  !store seed values, converting
    xFN = $CVTMISS(First_Name,spc), ! missing to spaces
    xMN = $CVTMISS(Middle_Name,spc),
    xId = $ID,                      !set record pointer
    lastWasDup = FALSE,             !initially no matches
    dupsCount = 0;                 !count of unique matches

GETRECORD                          !position first test record

REPEAT                              !start of comparison processing

IF xLN EQ $CVTMISS(Last_Name,spc) AND - !compare stored record
    xFN EQ $CVTMISS(First_Name,spc) AND - ! to CURRENT record
    xMN EQ $CVTMISS(Middle_Name,spc) -
THEN                                !ismatch
    IF NOT lastWasDup THEN          !if first match,
        LET lastWasDup = TRUE,     ! set stored match flag,
            dupsCount = dupsCount +1 ! increment dup counter
    END_IF

ELSE                                !notmatch
    IF lastWasDup THEN             !if last stored was matched
        LET lastWasDup = FALSE     ! indicate unmatched
    ELSE
        DROP $ID xId              ! or remove unmatched
    END_IF

    LET xLN = $CVTMISS(Last_Name,spc), !reset stored record
        xFN = $CVTMISS(First_Name,spc), ! for next possible
        xMN = $CVTMISS(Middle_Name,spc), ! match
        xId = $ID;
END_IF

```

(continued on next page)

(Figure B continued)

```
GETRECORD ON_END                !advance to next record
  LET noMoreRecs = TRUE          ! mark for end-of-set
  END_ON

UNTIL noMoreRecs                !end processing

IF NOT lastWasDup THEN          !if last record was unmatched
  DROP $ID xId                  ! drop it
END_IF

PRINT dupsCount TITLE "# Distinct Duplicate Records"
SHOW SELECTION_SET
END
```

Explanation and High Point - DROP \$ID

Upon the first matching record, the flag lastWasDup is set, as there may be multiple matches for a datum, and the uniqueness counter dupsCounter is incremented. On the first non-matching record, we note this fact by resetting the match flag, then storing the new record's values, and then on to the next following record. On the next non-match, we check the match flag: if it was not set, we've had two non-matches in a row, so the stored value doesn't match anything. We drop the record from the selection set via the DROP \$ID command. DROP \$ID processing lets us remove a non-current record from the selection set.

Processing continues in this manner until the end of the selection set, where we test the match status flag for the last record to see if it had a match. If not, again, it's dropped from the selection.

At the end, we have a counter showing the number of unique duplicates, as well as leaving the selection set in sorted, matched order.

Summary

While FIND DUPLICATES processing cannot operate upon multiple attributes, it is still possible to do so programmatically, via a variety of methods. If there are enough keyed attributes to be matched, selection set manipulation can be used to perform the proper matching. A simpler approach using value matching can also be done, as described in MULTIDUP.DMC.

© 2006 Computer Corporation of America
500 Old Connecticut Path, Framingham, MA 01701