Issue Date: FoxTalk May 1997

A "UNIQUE" Approach to De-duping in FoxPro

Dave Jinkerson and Bob Grommes

Here's a simple and elegant approach to a seemingly difficult problem: removing duplicate records from a table.

Have you ever received two or more copies of the same mail piece from an advertiser? Ever had an incorrect black mark on your credit history purged, only to have it show up again weeks or months later? These and other typical gaffes aren't annoying just to consumers, but they also cost businesses untold millions in wasted effort and postage.

These things happen because many companies fail to properly manage their consumer databases. One of the most important management issues for name and address data is name and address de-duping -- that is, removing duplicate instances of the same address from a database.

To de-dupe or not to de-dupe?

You might argue that de-duping isn't necessary, provided that you never allow duplicates to be entered in the first place. But in the real world of database marketing, address records come from many sources -- outside mailing list purchases and imports from sales leads' databases, for example. What you need is a way to append records from any number of independent sources and quickly remove the duplicates. Often, for best results, a set of imported records should be processed with address standardization software and de-duped against itself before being added to your master table and de-duped against that.

Here at Younger Direct Software Solutions, we manage large databases, primarily using Visual FoxPro and Oracle, for clients in the leisure and recreation industries. The larger consumer tables have upwards of 2 million names and addresses, and we're expecting much larger tables in the near future. These tables are de-duped constantly against themselves and against outside sources. We needed a way to do this as quickly as possible, because turnaround time for new data is very important to our clients.

While the system we eventually designed is faster and more elaborate (and proprietary -- read "top secret") than what we describe here, the following technique is extremely simple, very fast for tables with up to about 400,000 records, and is applicable to a broad variety of de-duping situations. Also, it provides an additional use for the much-maligned "unique" style index, and showcases the advantages of FoxPro's unique expression-based indexes.

In search of the perfect match

De-duping is always based on a matchkey -- an indexable code or expression. All records with the same matchkey are considered to be identical for de-duping purposes. For example, a simple temporary name and address matchkey index for U.S. addresses would be constructed like this, presuming that name and address data are stored in uppercase:

```
INDEX ON LEFT(zip,5) + LEFT(address1,10) ;
  + LEFT(lname,8) TO DeDupe COMPACT
```

Many of us who work primarily with FoxPro forget how powerful FoxPro's expression-based indexes are. If we used the lesser capabilities of other databases, we'd only be able to index on one field or on simple concatenations of multiple fields. If we needed more complex index keys we'd have to create a field to hold them and write code to maintain them. Expression-based indexes really shine when it comes to matchkeys; you can create matchkeys, even "permanent" ones, based entirely on indexes.

Because ZIP codes define the city and state, an address matchkey need only include the first five digits of the ZIP code, and doesn't need to include city and state. This also gets around variances in how city names are spelled and the possibility of "vanity" city names. For example, residents of suburban towns often give out the name of the larger nearby city because it's a legal alternative address and the city name is better known than the suburb name. By using the ZIP code *instead* of the city and state in the matchkey, two addresses for Joe Doakes at the same street address in Phoenix and Glendale, Arizona, will still be considered duplicates, because Joe will have given out both addresses with the same Glendale ZIP code.

Of course, this and any other address matchkey is only as accurate as the address, so you should use standard tools such as ArcList and AccuMail to standardize and correct the data before developing a matchkey and performing a de-duping operation.

Another feature of the previous matchkey is that only the first portions of the address and last name fields are used. This helps get around minor variances in spelling and punctuation. The idea is to make the key wide enough to be meaningful but narrow enough to be forgiving. Our analyses have shown that for typical domestic names and addresses, the first 10 characters of the primary address line and the first eight characters of the last name are the "sweet spot" that yields the best overall results.

An even better version of this matchkey removes the ambiguity introduced by punctuation and white space:

```
INDEX ON LEFT(zip,5) ;
```

```
  + PADR(CHRTRAN(Archive.addr1,',.-#&()/\ ',''),10) ;
  + PADR(ALLTRIM(Archive.lname),8) TO DeDupe COMPACT
```

Here we've used CHRTRAN( ) to remove punctuation (including spaces) from the address to decrease the likelihood of a mismatch based on insignificant punctuation differences, then used PADR( ) to make sure the resulting key fragment is of a constant length. *Warning:* do not use LEFT( ) in a matchkey expression; it sometimes produces inaccurate results, particularly if you later use the same expression to relate another table to this one by matchkey.

If your name and address data aren't stored in uppercase, you'll want to uppercase the whole index expression:

```
INDEX ON UPPER(LEFT(zip,5) ;
  + PADR(CHRTRAN(Archive.addr1,',.-#&()/\ ',''),10) ;
  + PADR(ALLTRIM(Archive.lname),8)) TO DeDupe COMPACT
```

A final consideration involves your business rules for this particular table. Do you want to send more than one mailing piece to each household, in the event that you have addresses for more than one person at that address? If you want to de-dupe actual people, rather than families, you should include, say, the first five characters of the first name in the matchkey as well.

Other possible considerations: If you're doing a "Joe Doakes or current resident" type of mailing, you might leave name data entirely off the matchkey. If you have really "clean" data, or reliable and standardized extra fields such as titles (Mr., Mrs., and so on), it may make sense to include those in the matchkey too. To develop a good matchkey, you have to know your data and think and experiment a bit. We've done a great deal of research, and one thing it has told us is that *there is no such thing as an "ideal" or "universal" matchkey.* Matchkeys must be matched to the data and to your immediate purpose, and they need to be re-evaluated from time to time.

Doing the deed

Okay, so you've developed a matchkey. Now what? You aren't going to believe how simple it is. The following five commands will open and de-dupe any table, cTableName, based on any matchkey, cMatchKey, using the temporary index specified in cTempIDX:

```
USE (cTableName) EXCLUSIVE
DELETE ALL
INDEX ON &cMatchKey UNIQUE TO (cTempIDX) COMPACT
RECALL ALL
CLOSE INDEX
```

What's going on here? DELETE ALL will mark all records in the table for deletion. The UNIQUE index will *create a key only for the first occurrence of each unique matchkey.* Thus, once the UNIQUE index is active, doing a RECALL ALL will recall only the records corresponding to those unique keys; all the duplicates will remain deleted. You can now filter them out with SET DELETED ON or by PACKing the table.

Another beautiful thing about this technique is that it's reversible. Right after the CLOSE INDEX, you can Index on the matchkey again without the UNIQUE clause, make sure DELETED is set OFF, and view the table in matchkey order to see if you got the intended result. If you want to compare a different matchkey, you can undo the de-dupe with another RECALL ALL. Remember that as long as the UNIQUE index isn't active, RECALL ALL will un-delete everything and you can have another go at the same table.

Obviously, this technique presumes that there are no deleted records in your table that you want to retain. The table you're de-duping must be "clean" in this regard. It really doesn't matter, however, if there are existing deleted records merely because of a previous de-duping pass, because given the same matchkey, your new pass will leave all those records deleted again in the end anyway.

Limitations

RECALL ALL is the bottleneck of this technique; it takes progressively longer with larger numbers of records. It will work correctly on a table of any size, but on larger tables (more than 400,000 records) it sweeps through the table more and more slowly as it executes. We've found even faster techniques that work faster on large files -- techniques so slick that they're a corporate secret. We could tell you about them, but then we'd have to kill you! However, if you need a "black box" routine to work on larger data sets, we've been known to do outside contracting -- send us e-mail or call us.

Limitations aside, this technique is simple, fast, and effective. Keep it in mind, not just for name and address de-duping, but for "tiny" applications, such as removing duplicates from a combo box list source.