

# What's in That Data Set?

Whil Hentzen

Most Fox developers these days are running into the situation of being asked to take over another system. One of the first things you'll need to do is evaluate that system. A previous article described a tool that gathered a variety of statistics about the code in a system and then discussed what to do with that information. But a system isn't just about the code. This article describes a tool that helps you quickly get your arms around a data set.

Data in Fox applications comes in several flavors, from plain DBFs to DBCs and their related files to SQL databases. This article addresses the simplest, and likely most common, of those scenarios - a collection of DBFs in a folder.

When faced with a folder full of DBFs (and their associated files, such as FPTs, CDXs, and perhaps IDXs), the inclination is to open up a few tables and see what they look like. However, for all but the smallest system, that approach will give you only a pinhole's view of the data, and as such, likely an unreliable one.

Better to get a consistent and comprehensive view of the data set. What might that view consist of?

## A Comprehensive View of the Data Set

Well, first off, that view should include a list of tables, together with basic statistics, such as size (in bytes), number of fields (how wide), number of indexes, and number of rows (how long). These stats give you a big picture of how many tables there are and which ones are the major players. It may also give you an idea of what the tables are used for. A table with dozens of fields and tens or hundreds of thousands of rows is likely to be a transaction table, while a table with a few fields and a couple dozen rows is probably a look-up table.

Running the form DataSet.scx, selecting a folder, and clicking the Analyze button displays such a list in the Files tab, a birds-eye view of the DBFs in the folder, as shown in Figure 1.

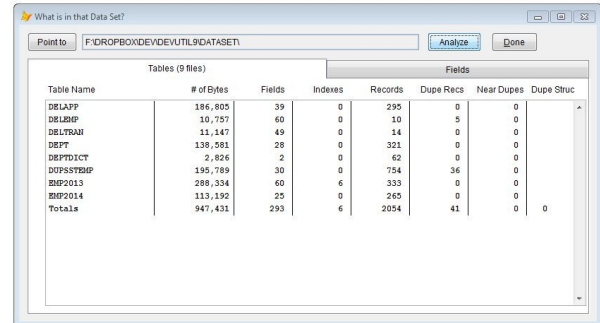


Table Name	Tables (9 files)			Fields			
	# of Bytes	Fields	Indexes	Records	Dupe Recs	Near Dupes	Dupe Struc
DBLAPP	186,605	39	0	295	0	0	
DBLRBP	10,757	60	0	10	5	0	
DBLTRAI	11,147	49	0	14	0	0	
DBPF	138,091	28	0	321	0	0	
DEFPDICT	2,826	2	0	62	0	0	
DOPSTEMP	195,789	30	0	754	36	0	
EMP2013	288,334	60	6	333	0	0	
EMP2014	113,192	25	0	265	0	0	
Totals	947,431	293	6	2054	41	0	0

Figure 1. The Files tab.

So within a few minutes, you can get an idea of what types of tables are in the system and what roles they might play.

In this same view, I've added a couple more columns; but these aren't raw stats; their purpose is to help you gain insight into the system. The first is the number of duplicate records in a table. While there may be legitimate reasons for an entire record to be duplicated, more often it's a sign of potential trouble. Duplicate records probably indicate mistakes in data entry or processing, or incorrectly normalized data. This type of data could generate false test results as well as bad information for the user.

A row doesn't have to be exactly identical in order for it to be bad. For instance, if the same record is saved twice, it could have different time stamps, but the rest of the fields would be the same. Thus, I've added a column for future use, that to identify number of 'near duplicates' - how to define a 'near dupe' is an exercise left for the reader in their particular situation.

And the last column in our birds eye view flags the number of tables that have identical structures. Like the duplicate rows scenario, there may well be scenarios where it's OK for two tables to have the same structure, just as often this may be a warning flag. Bad table design, tables created and then abandoned, but not deleted (e.g. invoice.dbf, invoice1.dbf, and invoice2.dbf), or non-normalized data could all result in tables with duplicate structures residing in the same folder.

## A Granular View of the Data Set

Once you've gotten the big picture, it's time to look at the data set more granularly.

The Fields tab in the Data Set form drills down into table/field combinations, with a second page frame that contains five tabs.

### Empty fields tab

The first tab, Empty, displays how many fields are completely empty, that is, bereft of any values, as shown in Figure 2.

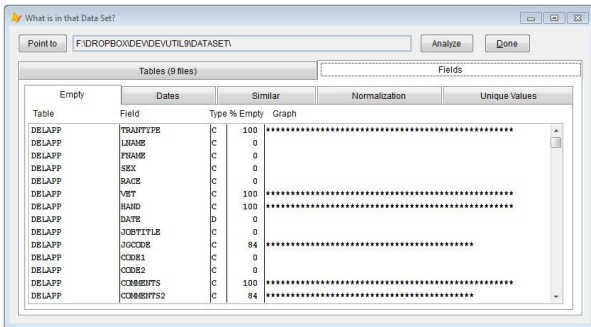


Figure 2. The Empty tab.

If a field has NO values.... why is it in the table at all? Either it's a symptom of bad design, or, worse, there's supposed to be data in it, and some errant process has wiped it all out. The list shows every table/field combination in the data set, the data type, percentage empty, and includes a simple bar graph so you can easily spot the miscreants.

### Date ranges tab

The Dates tab, shown in Figure 3, does an analysis of all date and date/time fields in the data set. Starting with the current decade, the distribution of dates across the years is calculated. The combo box can be used to change which decade the distribution is calculated over. Date fields that are heavily loaded with dates in certain ranges can be a clue that the use of the field, and perhaps the table itself, has changed over the years. Alternatively, that could be a clue that there are program errors that are causing dates (and possibly other data) to no longer be saved.

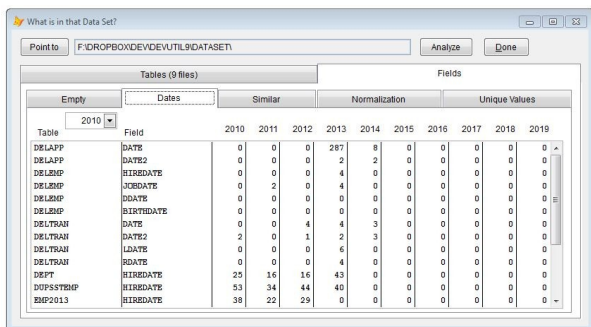


Figure 3. The Dates tab.

### Similar field names tab

An examination of field names in a table can show you how well the data is normalized. Fields with names like date1, date2, date3 or phoneA, phoneB, phoneC may be an indication that the developer didn't think through the design very well. (Of course, it may also be an indication that the requirements of the system have changed over the years, and additional fields were added later, despite proclamations of the user that 'we will NEVER have more than Phone field in the table.')

The Similar tab shows all fields, and for each field name, how many other fields in the table use that field name as a stem. See Figure 4.

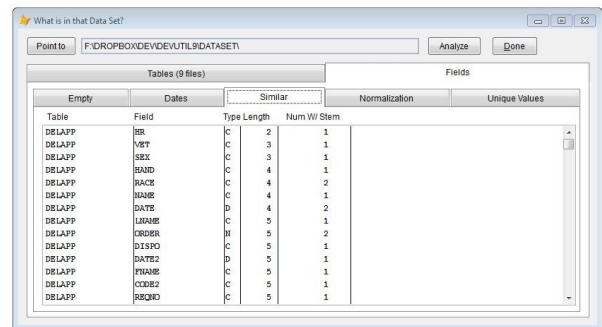


Figure 4. The Similar tab.

### Normalized field names tab

Another type of field name examination involves looking for field names that are used across tables. Compared to other data auditing procedures, field names that are used in more than one table can be a good thing. Data purists believe that the same field name should be used for the same data element, regardless of where it's found in the data set. (Old timers may remember the standard of including a table identifier in the field name; I was never a fan.)

The Normalization tab, shown in Figure 5, displays each field name along with which tables it's found in.

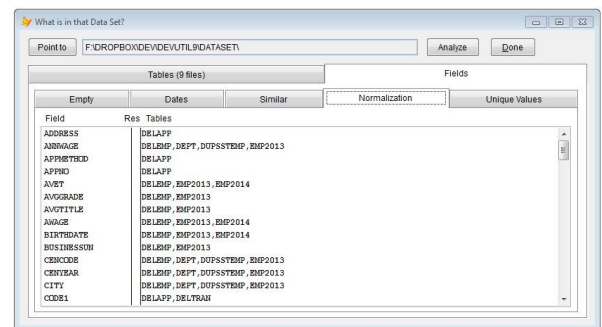


Figure 5. The Normalization tab.

## Unique Values

One final piece of data that may be useful is how many unique values are found in a field. Figure 6 shows the Unique tab, where each field is listed, together with how many rows and how many unique values for that row exist.

Empty	Dates	Rows	Similar	Normalization	Unique Values
		295			
DELAFF.DBF	PREMIUMTYPE	295			1
DELAFF.DBF	LIQUID	295			283
DELAFF.DBF	FINAME	295			214
DELAFF.DBF	SEX	295			3
DELAFF.DBF	RACE	295			5
DELAFF.DBF	NET	295			1
DELAFF.DBF	HAND	295			1
DELAFF.DBF	DATE	295			92
DELAFF.DBF	DOBTTITLE	295			34
DELAFF.DBF	LOGCODE	295			2
DELAFF.DBF	ICD9I	295			8
DELAFF.DBF	ICD9E	295			9
DELAFF.DBF	ICD9BIRTHS	295			1
DELAFF.DBF	ICD9BIRTHS2	295			17

Figure 6. The Unique Values tab.

Naturally, the purpose of a field will have an impact on the data in it, but even without an explicit description, you may be able to spot anomalies. For instance, an "InvoiceDate" field should have a fairly broad range of dates; if there are only a few, this might spell trouble. On the other hand, a look-up table, such as "State" should have a fairly small and well-defined domain. If the "State" field contains hundreds of values, it could very well contain a variety of spellings and codes for the same value, which would make reporting and searching on that field unreliable. The bottom line, is the domain for each field well-defined, or are there lots of near misses, indicating that standard data entry and validation is not enforced?

## What To Do With The Results

Now that you've got all this data, what do you do with it?

The general process will be to look problems, ask the user for explanations of unexpected values, look at the code for problems of values that weren't explained by the user, and finally, save off the data from this analysis and take regular snapshots for comparison, to see if the identified problems get better or worse.

Let's start with the first tab and look at what values might be unexpected, and discuss some options.

Actually, let's start even earlier.

Before you look at the Files tab, open up Explorer or whatever file manager you use, and sort the list of DBFs by last date modified. Files that haven't been changed in a long time ("long" being a subjective term depending on the system) may either be static (the "States" table with a last modified date of 1994 is perfectly OK) or may

indicate an unused table (a health premiums table that was last touched 15 years ago is unquestionably out of date and thus has likely been superseded by a newer version.)

Now lets look at the tabs.

## Files tab

The unusual values to look for are any duplicate rows, duplicate structures, and, if you've added code to find near dupes, those.

## Empty tab

Obviously, any field that is 100% empty is a candidate for concern. What's more amorphous are fields that are almost completely empty. Suppose a ten thousand row table had a field where only seven rows contained data. There could be a reasonable explanation, but it's worth investigating.

## Dates tab

The values in the Dates tab represent the number of rows that contain a date for each year in the decade of columns. So suppose in a table there are a dozen rows that have a transaction date in 2010, another dozen rows with a 2011 transaction date, and so on, where, to date, there are 4 rows with 2015 transaction dates. The columns would look like this:

```
2010 2011 2012 2013 2014 2015 2016 2017..2019
    12  12  12  12  12  4  0  0.. 0
```

There are two unusual patterns of data to watch for. The first is a bulge of values that should be evenly distributed, like so:

```
2010 2011 2012 2013 2014 2015 2016 2017..2019
    2  451 399 502  1  4  0  0.. 0
```

Why did the number of rows bulge for three years and then drop again?

The other unusual pattern is an uneven distribution, like so:

```
2010 2011 2012 2013 2014 2015 2016 2017..2019
    2  451 399 502  17 197  0  0.. 0
```

Why did the number of rows with dates skyrocket in 2011 and then plummet in 2014, only to boost up again the next year? Worth asking.

## Similar tab

As noted earlier, multiple fields with the same stem might indicate poorly thought out data design or repeated changes to the data structure (and thus possible tenuous modifications to the

code.) If the fields have different data types and/or lengths, however, a simple similarity in names is probably not a concern.

### ***Normalization tab***

This tab contains information that is more helpful in understanding the data design rather than warning about problems. It can be useful to see where similarly named fields are found throughout the data set.

### ***Unique Values tab***

The Uniques column contains the number of unique values for a field. In some fields, this number should be low, such as gender or type. Other fields should have a large number of unique values, such as last name. And a few fields will likely have the same number of values as rows, such as invoice number and the field that contains the PK.

Familiarity with the fields together with eyeballing the rows vs uniques is really the only way to detect anomalies that should be investigated.

## **What To Really Do With The Data**

Just like my last article, the primary purpose of this analysis is to become informed about what you're getting into. A secondary goal is to be able to explain to your customer what possible situations you might run into.

### **Author Profile**

*Whil Hentzen is an independent software developer based in Milwaukee, Wisconsin (as opposed to Milwaukee, Minnesota, as many people think.) His writing has killed many trees over the years, but none since 2007. He has realized he really sort of misses it. You can reach him at [whil@whilhentzen.com](mailto:whil@whilhentzen.com)*