

Merge Database Records, by Mark Stefanchuk

Published in Control-Alt-Delete Magazine, Q3 2000, by Pen & Brush - Australia

With a little ingenuity we can simplify anything. At least that's what I thought. And I still do. But, as I learned, discovery takes time. In the end though, when the time to complete a task is reduced from weeks to seconds the extra effort is always worth doing.

That's what this article is about, a MicroStation basic macro that in less than 100 lines eliminates the need for duplicating data entry. There are some bonuses too, like improved drawing quality and data integrity.

Merge Database Records is the result of more than a year of effort. It represents an attempt to capture facility-move-data for a large manufacturing client. Here's some background. Updates of our client's facility-moves-records, including database and drawings were out sourced to a local architectural firm. What was the problem? The turn around time for updated drawings was six months. That time frame was much too long to accurately track moves and they didn't like the cost of the service. They decided to look for an alternative.

Our client decided that there were three choices. The first choice, out source the work to another architectural firm. The second choice, purchase the same facility management software that the architectural firm was using and do the work internally. Neither of these two choices worked for them. If they out sourced to another firm then they would have the same expensive system and would continue to receive out dated drawings. The second choice required intensive training on new and sophisticated software. Each designer has several daily tasks, and CAD is not their primary function, so adding hundreds of new features coupled with the overhead and maintenance of an Oracle database didn't make sense. So, they needed a third alternative. That's where I come in. Someone said, lets ask Mark.

What I decided upon were three applications. Two of them they owned and were familiar with, MicroStation and Microsoft Access. I suggested alternative databases but they wanted a tool that the designers had used and would have the ability to administer themselves. The third application is dbWindow from Next Level, Inc. The application is a generic database application written in MDL for MicroStation. It adds several features for data manipulation that MicroStation lacks. The cost per seat for this solution was less than \$1000US. That included both dbWindow and Microsoft Access. The cost of the high-end facility/asset management software used by the old architectural firm is more than \$30,000US. So great, I saved them some money.

The idea was to keep the solution simple. This drove our decision more than the cost. We just wanted a way to add new occupants to office space and create a report or two. The commands also needed to be simple so that they would be easy to remember. With training, and some initial consulting they were ready to go. Adding, and moving occupants worked as designed. But, there were some problems.

Since they didn't have a complete understanding of how MicroStation interacts with databases, the displayable attributes (text nodes with database linkages) were moved and copied using MicroStation commands instead of edited with the dbWindow. Next, the ODBC versions on several computers were not up to date which meant that the database and drawings were not synchronizing correctly. We fixed these problems but there was still a bigger issue. The time required to update move data from day to day was taking too much time. The data entry was not accurate. We needed to make some changes.

The first thing we tried was to build some tools that updated and removed records automatically. The idea was that if we could build tools to reduce some steps when occupants were moved or new employees were added that we could speed up the process and reduce some mistakes. Unfortunately, we decided that this wasn't enough. We also learned about this time of a second Microsoft Access database used by the electrical maintenance department. This additional database tracked phone and data drop requests and it so happened that the facility management department initialized the move request with this database. What the new database meant to us was that we could get all of the updates for our facility plans by synchronizing this database with our drawing database.

This was fantastic because now we didn't have to do data entry any more. No moving, no copying, no data entry, no kidding! When we set up our facility plans we decided to use displayable attribute text nodes. We did this so that we could edit the database and use the features of dbWindow to update the graphics automatically. And with the electrical maintenance database we had uncovered a way to get the data we needed without manually editing our database. All of the data entry would occur at the time a move was requested.

So, lets get down to business. We want to transfer records. We just have to figure out how to do it. I decided to use a MicroStation basic macro, mainly because it is very simple to implement and deploy.

In our macro example we will synchronize first name, last name, organization number, and telephone number. The merge will be based on a move completion status, and the room number. I have simplified the databases and their tables for this discussion, but it is easy to extend the concepts to larger and more sophisticated tables. Lets look at the tables.

Employee.mdb is the Microsoft access database used to manage employee locations. Notice that the employee table contains an MSLINK field, a requirement for linking data to MicroStation graphics. Also notice that two other tables are included in this database. They are mscatalog and msforms. MicroStation requires these tables so that it knows which tables it is allowed to modify.

The employee table includes several records. Some of the records show employees occupying spaces and others indicate vacant rooms. The vacant rooms have room numbers that match room numbers in the moves database.

	LAST_NAME	FIRST_NAME	ORG	PHONE_EXT	ROOM	MGRDPT	MSLINK
▶	CRAIG	JENNY	BB100000	2345	101	BB100000	1
	SMITH	FRED	BB100000	2356	102	BB100000	2
	GORE	AL	BB100000	2357	103	BB100000	3
	BUSH	G. (SHRUB)	BB100000	2358	104	BB100000	4
	VACANT				244		5
	VACANT				245		6
	VACANT				246		7
	VACANT				247		8
	VACANT				248		9
*							0

Record: 1 of 9

Figure 1: Employee Table

Moves.mdb is the Microsoft Access database used to initiate move requests. It does not contain an MSLINK column in the moves table and therefore cannot be linked to graphic elements. Notice though, that the moves database does contain mscatalog and msforms tables. Without these, MicroStation cannot connect to the database and therefore cannot read the data inside them. So, if we want to use a basic macro to read a database we need to include both of these tables even when the data will not be linked to graphics. The merge records macro will never modify this database. It belongs to someone else. All we want is the data from active moves. The table has 6 data fields (columns), first name, last name, phone, date, organization, existing location, new location, and status. We need the data from all of these fields except date.

	Last Name	First Name	Work Phone	Date Received	Organization	Existing Location	New Location	Status
▶	Letterman	Dave	1236	8/14/2000	AB120000	305	247	Not_Completed
	Lenno	Jay	1235	8/12/2000	AB120000	304	246	Not_Completed
	Jones	Jenny	1234	8/8/2000	AB120000	303	245	Not_Completed
	Smith	John	1233	8/4/2000	AB120000	302	244	Not_Completed
	Bentley	Keith	1237	8/14/2000	AB120000	306	248	Not_Completed
	Stefanchuk	Mark	2000	7/1/2000	AC100001	1101	1402	Completed
	Ramsey	Mary	2001	7/1/2000	AC100001	1102	1403	Completed

Record: 8 of 8

Figure 2: Moves Table

Interfacing between MicroStation and a database requires a connection. Even if there is no interaction between graphics and the database, a connection is still required.

There are a couple of ways to connect to a database in MicroStation. The first method is interactive. Start MicroStation and enter a design file. From the Settings pull down menu choose Database->Connect. Since we are using Microsoft Access databases choose ODBC as the Database Server. In the Connect String field type in the name of the ODBC Data Source.

“What is an ODBC Data Source?” And, “Where did it come from?” In NT4 ODBC data sources can be easily added using a convenient ODBC icon located in the control panel. In Windows 2000 the tool is hidden away in the winnt directory and is called C:\WINNT\system32\odbcad32.exe. I made a short cut and put it on my desktop. Double click on this file to set up an ODBC Data Source.

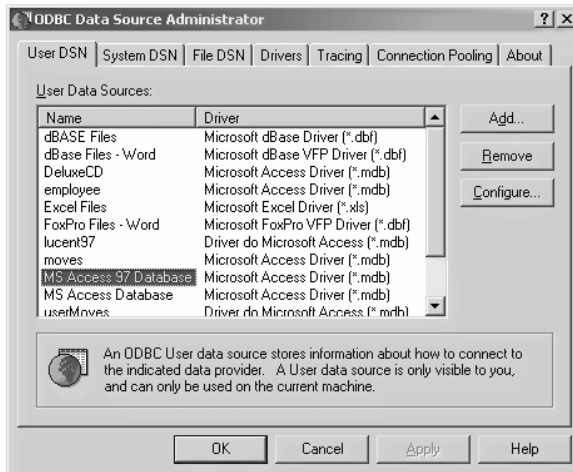


Figure 3: ODBC Data Source Dialog

Under the Name column click on MS Access 97 Database (or MS Access Database for the Access 2000 version) then click the Add... button located in the upper right corner of the dialog. On the next dialog click on Microsoft Access Driver, then click finish.

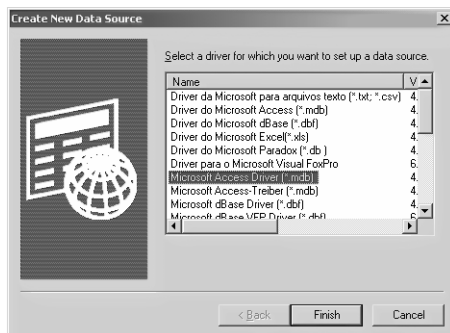


Figure 4: Create A New Data Source Dialog

Finally, you will be prompted to Set Up the new Microsoft Access data source. Type in a data source name. This will be the name that gets typed into the Connect String field in MicroStation.

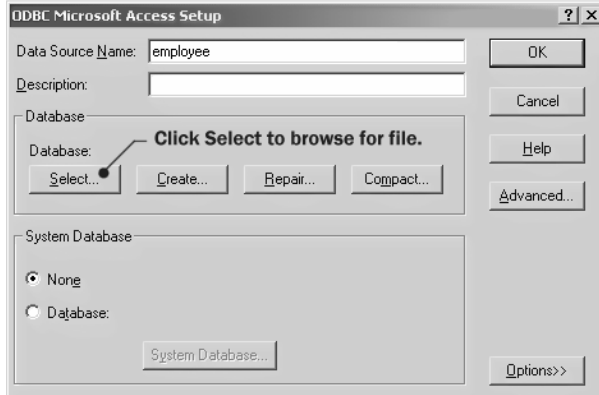


Figure 5: ODBC Microsoft Access Set Up

To finish the set up click on the Select... button in the Database group box and browse to find your database. Click Ok to end it all.

Alternatively you can execute the connect statement with a command line option.

Creating a desktop icon with the following target will allow you to connect to a database from a macro or other MicroStation application without interactively connecting to a database.

```
C:\WIN32APP\msse\ustation.exe -wdODBC
```

Copying the data from one database to another can be accomplished with just a few lines of code. There are two databases to connect to and two tables manage with the macro. The complete code with variable declarations is available on cadgurus.com. The first important piece of code is the connection to the moves database. Only the ODBC Data Source is needed and not a complete file specification.

```
dbStartMove.name = "moves"
dbStartMove.connect "moves"
```

Next the moves table is identified and a record search is setup. The status column is checked for records that need to be processed.

```
tblUserMoves.name = "moves"
tblUserMoves.criteria="where Status='Not_Completed'"
```

MicroStation uses MbeSQLda as a container for record data found by a search. The object has uses an array called value to hold the field data. Each index corresponds to a column (a field) in the database.

```
If tblUserMoves.recordFirst(sqlda)=MBE_Success Then
```

Use the do loop to step through each record found by the query. By the way, the query was created by the statement `tblUserMoves.criteria="where Status='Not_Completed'"`.

When MicroStation connects to the employee database the moves database will disconnect. Because of this, string arrays are used to store the move data.

Do

```

fn_m(m) = sqlda.value(1) ' first name
ln_m(m) = sqlda.value(0) ' last name
ph_m(m) = sqlda.value(2) ' phone
or_m(m) = sqlda.value(4) ' organization
el_m(m) = sqlda.value(5) ' existing location
nl_m(m) = sqlda.value(6) ' new location

m = m + 1
Loop while tblUserMoves.recordNext(sqlda)=MBE_QueryNotFinished
End If

```

The next step is to connect to the employee database and modify the records that match the moves. The macro looks for room number matches between the new location, nl_m(e), and the ROOM number found in the employee table.

```

dbManageMove.name = "employee"
dbManageMove.connect "employee"
tblEmployees.name = "employee"

```

for e = 1 to m

```
tblEmployees.criteria="where ROOM=" + "" + nl_m(e) + ""
```

The search criteria, tblEmployees.criteria="where ROOM=" + "" + nl_m(e) + "" finds the matching room numbers. There is only one record with the room number specified in the moves table so we only need to find the first occurrence of the room in the employee table.

```

If tblEmployees.recordFirst(sqlda)=MBE_Success Then
  mslinkval = val(sqlda.value(6))

```

The employee record is referenced by the mslink value and each column is updated with the corresponding string saved from the moves data base.

```

tblEmployees.recordUpdate mslinkval, "LAST_NAME=" + "" + ln_m(e) + ""
tblEmployees.recordUpdate mslinkval, "FIRST_NAME=" + "" + fn_m(e) + ""
tblEmployees.recordUpdate mslinkval, "ORG=" + "" + or_m(e) + ""
tblEmployees.recordUpdate mslinkval, "PHONE_EXT=" + "" + ph_m(e) + ""
tblEmployees.recordUpdate mslinkval, "MGRDPT=" + "" + or_m(e) + ""

```

End If

next e

Merging Database Records was not a difficult program to create, but uncovering all of the pieces to our client's facility moves problem took some time. Solutions don't always present themselves right away. So, don't give up. The difference between a good solution and a great one may only be a matter of time.