Managing Smart Devices Communications With A Database

Robert M. Orndorff
Dominion Virginia Power

Presented to the
Georgia Tech
Fault and Disturbance Analysis
April 26-27, 2004

Overview

This paper will present an introduction to databases and present some database design ideas specific to communications with substation IEDs. Also covered will be the implementation of databases used for storing data related to communications with IED (Intelligent Electronic Devices) at Dominion. There will also be discussion of how the use of these databases has made it very easy to maintain and distribute up to date phone and communications files that are used directly by the program performing the communications. This is meant as an introduction to database concepts and what can be accomplished using a database. The more intricate details of the database tasks will be omitted.

What is a Database?

Overview

For purposes of this paper, a database is a computer file, or group of files that holds data. The data can then be sorted, grouped, summarized, and reported on. A single table, such as a spreadsheet can be a database. A collection of many tables with related information can also be a database.

A table will have rows and columns. Each row is considered a ***record***. All data in a given row is related and should always be stored together. The columns are considered ***fields***. Each record (or row) may have many fields.

Single table

Probably the most common database used today is a spreadsheet. Spreadsheets make data entry and printing relatively simple and straightforward. A simple database consists of one or more records, and each record contains one or more fields. It is very important that all fields in a record stay associated with the proper record. Here is a simple address database.

Table 1

Name	***Location***
Clark Kent	Metropolis
Bruce Wayne	Gotham City

You will run into problems when attempting any complex functions on the data if it is in a spreadsheet program. Sorting, for example, if not done correctly will result in the data becoming corrupted. Some columns will be sorted while others are not. In the above example, if the name column is sorted and the city column is not, then your database will become useless.

A database program will perform the sorting correctly and keep all fields with their corresponding records.

Multiple table

When the data is more complex than the example above, you may find yourself entering the same information into many different records. Entering the same information multiple times is not only boring, but can lead to other problems such as typographical errors. All fields that should be the same may not be because of a mis-typed character. This can also be a maintenance problem if that data were to change at some point in the future. Every place that the repetitious data was entered will have to be updated to the new data. If a typographical error exists, then it may become difficult finding all the fields that need updating.

Here is a simple example of some repetitious data. In the real world, there are many more stations and each station is likely to have many relays. In this table, the station name and the phone number is duplicated.

Table 2

Station	***Relay***	***Phone Number***
Elmont	Line	555-2358
Elmont	Circuit	555-2358
Lakeside	Line	555-1235
Lakeside	Circuit	555-1235

This is where multiple tables would be useful. There could be a separate table for station names and numbers.

Table 3

Station ID	***Station Name***	***Number***
1	Elmont	555-2358
2	Lakeside	555-1235

Utilizing this new table to minimize data repetition in the relay Table 2 would work like this.

Table 4

Station ID	***Relay***
1	Line
1	Circuit
2	Line
2	Circuit

This table still has all the information contained in Table 2, but now some of the information is in Table 3 as well. It is very easy for a computer to “assemble” this information by reading the name and number from Table 3 and putting it with the relevant relay information. This assembling of data typically occurs in the form of a report. This also has the advantage of updating the name and number of every relay at a given station by changing the data at a single location.

This type of multi-table database is called a “Relational” database.

Designing a Relational Database

The design phase of creating a new database is generally the most difficult. It is not always easy to anticipate what your needs will be in the future. For example, when we created the database we currently use there was only the need for one phone number per device. Now, there may be a phone number as well as an IP address for a given device.

There is also the addition of communication port switches and possibly other means of communications in the future.

When designing a database, it is necessary to keep in mind the current and future uses of the data. For example, you may want to simply have a list of devices at any given substation so that when an operation occurs you can quickly determine if there is an IED that can supply you with data. You may also want to use the data to perform an autopoll. In this case it is necessary to have not only the phone (or IP) number, you would also need steering codes, port switch commands, passwords, and any other data needed to make the connection.

Some of the advantages of using a relational database are:
Minimizing data repetition - It is better to have the number stored in only one location in the database than in many.
Easier maintenance - By having only one filed to update, it is much easier to do and more likely to get done!
Better data accuracy - Reducing the number of entries also greatly reduces the chance of there being a typo in the data. If there are several fields containing what should be the same number and they don't all agree, it is then more troublesome and time consuming to determine which is correct.

Some Database Design Ideas for IED Communications

It is very important to do the best you can to plan for future needs when designing a database. A few concepts are presented here.

A Flatfile or single table database

A single table database can be very useful, however you can expect some repetition of data. The advantage of a single table database is that it's very easy to maintain and also easier for others (i.e. people that are not database experts) to use. Here is a listing of the fields we use in our database of relays.

Field Name	Data Type	Description
Station	Text	Substation name
Operating Number	Text	Operating number of device covered by this IED
Relay Type	Text	Relay model number
Performs Fault Location	Yes/No	Does this relay calculate a fault location
Fault Data	Text	Where does fault data go? Local or System Operations Center
Transmission	Yes/No	Yes if this is considered a Transmission asset
Interface Device	Text	Modem, comm processor, RTU, port switch
Phone Number	Text	Phone number
Passwords	Text	Passwords for access to this device
In Service	Yes/No	Is this device currently in service?
Notes	Memo	Any other notes about this device

This table layout will provide most of the necessary information to be able to know which devices to call and how to get the information out of them. There are drawbacks to this design, but they are manageable. For example, the station name and phone number are repeated for each device in a given station. The database program does have the ability to copy the previous record's data with a single keystroke. If the number or name changes, then each record containing that data will have to be changed. This can be manageable with the combination of search and replace and/or the "copy previous record" function. This table is lacking some information that is needed in today's environment. There is no field for IP address, for example.

One way to more efficiently store the same data is represented in Figure 1.

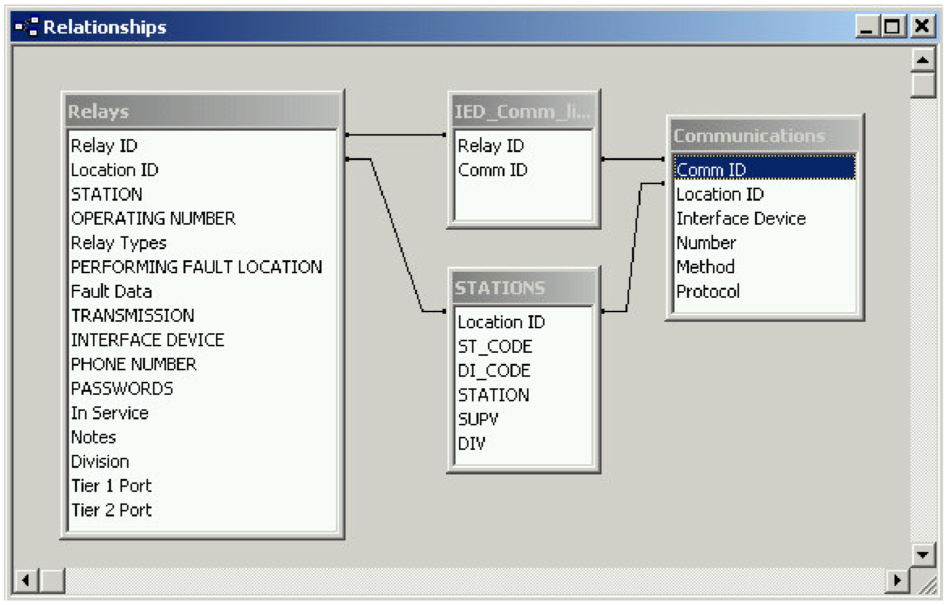

Figure 1 - One way to relate several tables

The same data is stored in this database; even the table titled "Relays" is very similar to the single table database example. The difference is that the phone number and station fields are no longer necessary. The relevant information is linked to other tables.

The new tables are "Communications", "Stations", and "IED_Comm_Links". Each record in the Communications table stores information about a communications method and is associated with a station. One communications method would be a dial up modem. Another would be a network connection. Now it is possible to have as many communications methods to a station as are needed. This would be very difficult to do with a single table database. An example would be a station that has a phone number and also an IP address. One station may have several phone numbers and IP addresses. Each record in the "Communications" table has a link to one station in the "Stations" table.

The "Stations" table has only a list of station locations and any other information relating to that particular station, such as division, district, or supervisor. Many records in other tables can be linked to a single record in this table.

The Location ID in the "Relays" table matches the Location ID in the "Stations" table. The name of the substation can be changed one time in the "Stations" table and all linked records will be linked to the new name.

The Relay ID and Communication ID link is a different story; there is not necessarily a direct one to one or one to many relationship. Because it is possible to have more than one communication record associated with a Relay **and** also possible that not all devices in a station will use the same communications method, there must be an inter-posing table that defines the links between relays and their associated communications records. This link table is called "IED_Comm_links". Each record in this table consists of just two fields - the link to the "Relays" table and the link to the corresponding record in the "Communications" table.

The fields that are common between tables that are used to link records from multiple tables are usually just a number. One way to create these numbers is to have the field defined as an "AutoNumber" field. The database program will automatically generate a unique number each time a record is created. These fields's data can usually be hidden from the person entering the data.

This multi table relational database seems like a very good way to store the data, and it is. However, there are obstacles to using this method. You must have someone that knows about database design to build it, or become a database expert yourself. It is also more difficult to find someone that can take over administration of the database should that become necessary. It is a little more of a chore to get all the data into the database initially, as you must take care to ensure that the links are correct. Once the database is set up, then maintenace of the data is much easier. Changes to the tables do require more work in a multi-table database.

Using the Data

Reports

The most common use of the data in a database is to run reports against it. This is a very useful tool because you can get some very specific information. For example, we have reports that specify transmission devices, distribution devices, devices that calculate fault location and send that data to the operating center, devices that calculate fault location and don't send the data to the operating center, and many more.

Scripting

One of the most powerful functions of a modern database program is the ability to do scripting, or programmatic control of the database. This has allowed us to easily maintain accurate phone files that are used directly by our communications program.

The evolution of this process started out of a desire to automate creation of Relay Gold scripts that we used to communicate with sequence of event recorders. We had a script for each recorder. Each script was virtually identical except for the phone number. This is when the automation began. A simple Quickbasic program was written that automatically created the scripts while it read the phone numbers from a text file. We then created icons (in Windows 3.1) that launched the script. Anyone could automatically download event data from SERs without having to know all the SER specific commands.

From this “one script per device” philosophy came the idea to have a single script template for each device, and have a program manage the scripting. This would reduce the need for many scripts and just create the one that is needed when it is needed.

As previously stated, this started out with just sequence of events recorders. When improvements were made to the SER script manager, there came a point that we realized that this method could work for any device that communicates via a terminal mode command line interface. That included not only sequence of event recorders, but also many types of relays, RTUs, communications port switches, and also some substation PCs. Once we realized the universality of this, we began the process of creating script templates for each type of device.

A little background

Most of the devices we communicate with have a command line interface, which means that we can use any terminal program. Windows Terminal, HyperTerminal, Procomm, Relay Gold, and Qmodem Pro are all terminal programs that can communicate with these devices. Many of these programs also support scripting. We were able to utilize all these scripting abilities to our advantage. We developed a program that writes a script file for the terminal program of our choice. This script is based on a template that contains connection instructions for that particular type of device. The template has a variable name in place of the phone number, so that when the “script organizer” program wrote out the communications script, the correct phone number was inserted. Other variables were also included, such as passwords, port numbers, etc. Once the communications script is written, then the communications program is opened; invoking the script.
Once this system was in place, it then becomes a chore to maintain the phone numbers for the script organizer program. This is where the database scripting is utilized. The script organizer program stores data in a standard text file that has the structure of Windows .ini files. Here is an example entry from a configuration file:

```
[ACCA 1]
Number=804-555-4575,,44,44,44,44
Type=SEL-2020
CallScriptTemplate=Q9600Sel.tpt
OutputScript=Autocall.qsc
WorkingDir=C:\Program Files\QmodemPro\Scripts
CallAction=C:\Program Files\QmodemPro\qmwin.exe autocall.qsc
Password=OTTER
Autopoll Template=sel-2020.tpt
Autopoll Commandline=c:\relay\relay.com autosel
Poll=Y
```

Since we store all of this information in a database, it seemed like a good idea to try automating the process of updating phone numbers. Also, because the script organizer program uses plain text files, it seemed that it would be relatively easy to write that type of file.

The database program that we use is Microsoft Access. There are several other desktop database programs available that will perform these same functions. Some others are Lotus Approach, Alpha 5, Filemaker Pro, and Paradox.

We were able to write a script in Access, which uses Microsoft's Visual Basic for Applications (VBA). If you are familiar with Visual Basic, or even earlier forms of BASIC, you should be able to recognize the commands used in VBA.

This script reads the data from the database and then writes it out in the format expected by the script organizer program. The VBA program uses a For..Next loop to parse through each record and then write the data to the specified file.

The code from the SEL-2020 export routine is shown in Appendix A.

This routine can be modified to match the format needed for almost any program that uses plain text configuration files. Binary files can also be written, it would just require a higher level of programming expertise. This same method could also just as easily be used to write script files for the communications program. For example, if you used Qmodem Pro, this routine could be modified to write the necessary data to a Qmodem script file that anyone could double click in Explorer and establish communications. Since the database program is writing the files, it's no big deal to write several hundred files in a few seconds. It is also much more likely that files will be kept up to date if it is very easy to do.

The advantage to all of this is that the phone number can be updated in only one location - the database. Then it is simply a matter of a few mouse clicks to have the data from the database in a format that is ready to be used to call an IED.

Another approach, in addition to the desktop database application, is to have the database on a web server and configured as an intranet web application. For most people this will require the help of a database expert from your IT department. Some advantages of storing these data on the intranet are:
-Better control of who can access the database
-Data can be accessed from any PC on the network.
-No special application required to be installed on the workstation.
-Different levels of access can be assigned on an individual basis, i.e. some people would have read only access while only a select few have full read and write access.
-Special scripting and reporting can be done from the web server, or a desktop database program like Microsoft Access can connect to the data on the server so that scripting and reporting can be accomplished locally without having your IT department write special routines.

Conclusion

Implementing a database to store and organize your IED communications information can make the job of communicating much easier and more efficient by having current data readily available. The process of updating communications files directly can be automated by utilizing the scripting functions of the database program.

References

Microsoft Corporation “Microsoft Access Users Guide”, ©1994 Microsoft Corporation

Alpha Software Corporation “Alpha Five for Windows Users Guide”, ©1994 Alpha Software Corporation

Biography

1.) Robert Orndorff works in the Fault Analysis group at Dominion Virginia Power and has held this position since November 1997. His responsibilities include:
-Retrieve and analyze data from DFR's, SER's, "smart" relays, and other devices for the entire Dominion system.
-Maintain, test and troubleshoot modem and network communications to substation devices
-Test and implement new technologies, including substation automation
-DFR Configuration and setup
-Anything else he's asked to do...

2) Robert previously worked as a field relay technician for 11 years. Responsibilities included installing, maintaining and testing of protective relay systems, SCADA, and power line carrier.

3) AAS Degree in Electronics from J. Sargeant Reynolds Community College. Hobbies include Amateur radio and computer programming.

Appendix A – Sample Microsoft Access VBA Script

```
Private Sub cmdExportPhone_Click()
On Local Error Resume Next
Dim A$, B$, X As Long, N As Long

CMDialog1.Flags = cdlOFNHideReadOnly + cdlOFNOverwritePrompt
CMDialog1.CancelError = True
CMDialog1.ShowSave
If Err.Number = 32755 Then
    Exit Sub
End If

A$ = CMDialog1.FileName
Dim MyDB As Database, MyRecordset As Recordset
Set MyDB = DBEngine.Workspaces(0).Databases(0)
Set MyRecordset = MyDB.OpenRecordset("SEL-2020", DB_OPEN_DYNASET)
MyRecordset.MoveLast
X = MyRecordset.RecordCount
MyRecordset.MoveFirst
Dim Poll As Boolean, Phone$, Telnet As Boolean

Open A$ For Output As #1
For N = 1 To X
    Phone$ = ""
    Poll = MyRecordset.Fields![In Service]
    Phone$ = MyRecordset.Fields![Phone]
    If InStr(1, Phone$, ".") Then
        'It's a telnet address
        Telnet = True
        Poll = False
    Else
        Telnet = False
    End If
        B$ = MyRecordset.Fields![Station]
        B$ = B$ + " " + MyRecordset.Fields![Unit]
        If Telnet = True Then
            B$ = B$ + " Telnet"
        End If
        Print #1, "[" + B$ + "]"
        If Phone$ <> "" Then
            B$ = Phone$
        Else
            B$ = "None"
        End If
        Print #1, "Number=" + B$
        Print #1, "Type=SEL-2020"
        If Telnet = False Then
            Print #1, "CallScriptTemplate=Q9600Sel.tpt"
        Else
            Print #1, "CallScriptTemplate=QTelnet.tpt"
        End If
        Print #1, "OutputScript=Autocall.qsc"
        Print #1, "WorkingDir=C:\Program Files\QmodemPro\Scripts"
        Print #1, "CallAction=C:\Program Files\QmodemPro\qmwin.exe autocall.qsc"
        Print #1, "Password=OTTER"
        Print #1, "Autopoll Template=sel-2020.tpt"
        Print #1, "Autopoll Commandline=c:\relay\relay.com autosel"
        If Poll = True Then
            Print #1, "Poll=Y"
        Else
            Print #1, "Poll=N"
        End If
        Print #1,
    MyRecordset.MoveNext
Next N
Close #1
MsgBox "Export Complete!", vbOKOnly + vbInformation
End Sub
```