DATABASES & DIGITAL FAULT RECORDS

Amir & Maria Makki, George Semati (SoftStuf, Inc)
Tony Giuliante (ATG Consulting)
Jeff Pond (National Grid)
Angela Rothweiler & Keith Pierce (PSE&G)

Submitted to the Fault and Disturbance Conference, April 2004

ABSTRACT:

Microprocessor based protection and measurement equipment such as digital relays and digital fault recorders (DFR) generate large volumes of data called digital fault records. Without the use of databases it would be near impossible to manage such volumes of records.

Historically, over the past 20 years, many types of fault record databases have been developed and many of them are still in use today. The databases were mainly based on the operating system's directory structures and file allocation tables. Recently, advanced database technologies are being explored. Such technologies include but are not limited to relational, object oriented and time series architectures (examples are PI, Oracle, Access, dBase, Paradox, Sybase and so on).

The paper provides a survey of the various types of database technologies old and new. Special emphasis is on databases of Comtrade records. References to the latest filing conventions are also made. The intent of the authors is for the paper to serve as a tutorial on the subject of databases and fault records.

DATABASES:

Database design (see Appendix A for definitions) is at the heart of all computer programs regardless of the application type. It is about organizing data in a related, logical way in order to simplify the task of finding and retrieving selected information.

Ever since the early days of computing, programmers have aspired to build new applications that automate or simplify the process of designing databases. Designing a database is not much different from writing a book. To write a book, you need to build a table of contents organized in parts, chapters, sections, and pages, and you need to provide a list of figures and tables and an index of terms and finally references. To design a database, you need to build a table of data structures organized in objects, fields, methods, and instances and you need to provide a list of menus and options and an index of functions, addresses and pointers. Again, the main objective is to simplify the task of retrieving desired information.

Historically, database design has been an art form. The organizational and relational design methodologies vary depending on the type of intended program or application. For example, assemblers have to organize machine code; compliers have to organize source code; operating systems organize files; object oriented and relational database programs organize transactions, operational data, and so on. The main question is: what is the best way to organize fault records?

DIGITAL FAULT RECORDS:

Before we address the above question let's examine the nature of fault records:

Fault records originate from various types of digital relays, fault recorders, meters, remote terminal units, and so on. In general, the records are saved in computer files and the formats are classified as either standard or proprietary.

There are many types of proprietary formats (ASCII or binary) in circulation today. The formats vary depending on the manufacturer and type of originating device. Figure-1 shows a few ASCII examples from various types of digital relays, and Figure-2 shows an example of a binary format from a DFR.

On the other hand, there are only a few standard formats (ASCII or binary) in circulation today and the most popular is the IEEE C37.111 COMTRADE standard. Figure-3 shows an example of the standard COMTRADE ASCII format and Figure-4 shows an example of the standard COMTRADE binary format.

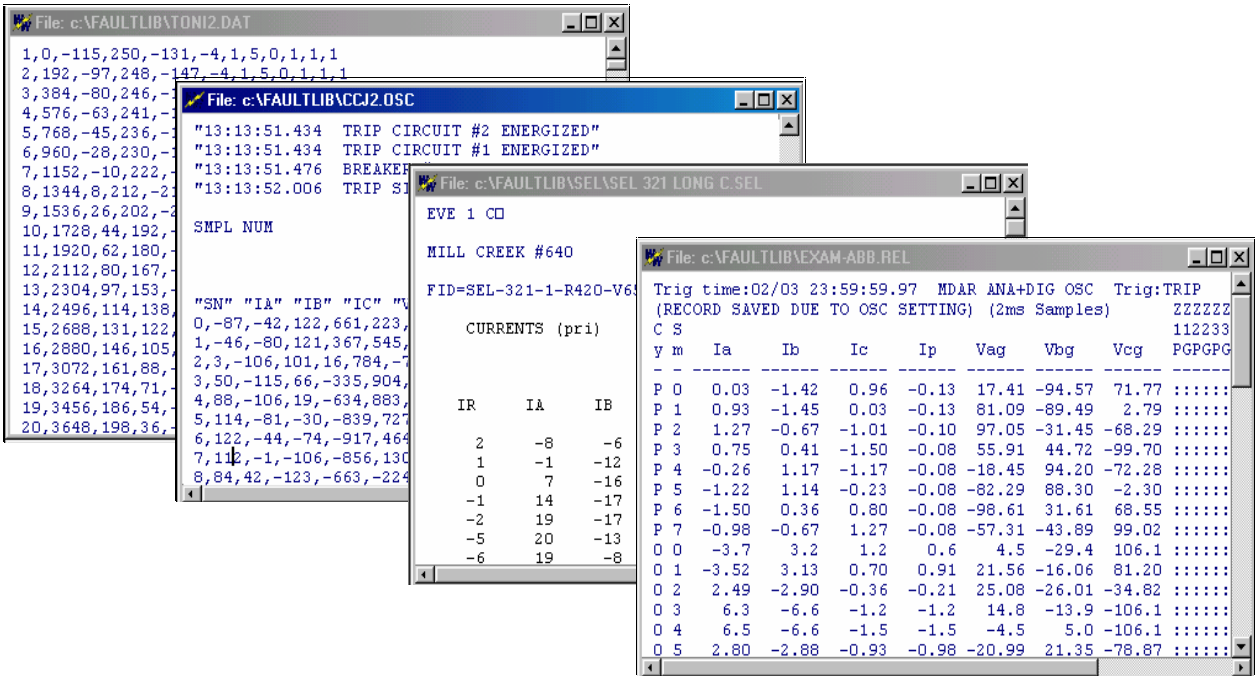


Figure-1: Various Types of ASCII Formats

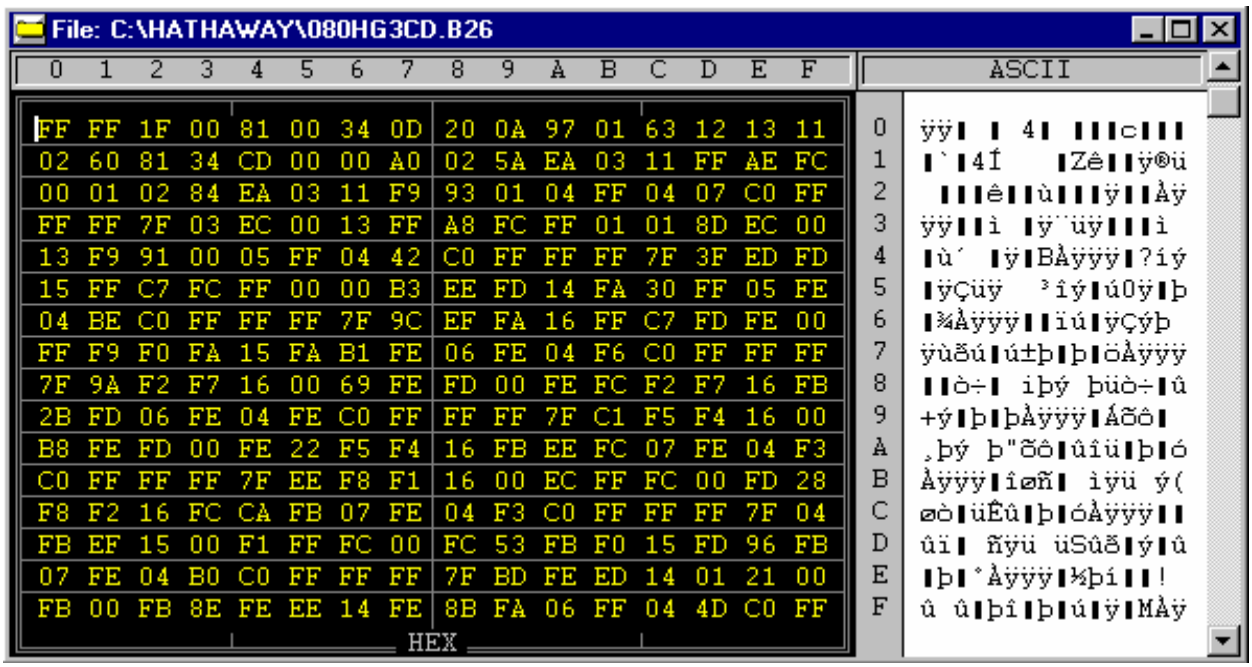


Figure-2: A Proprietary Binary Format

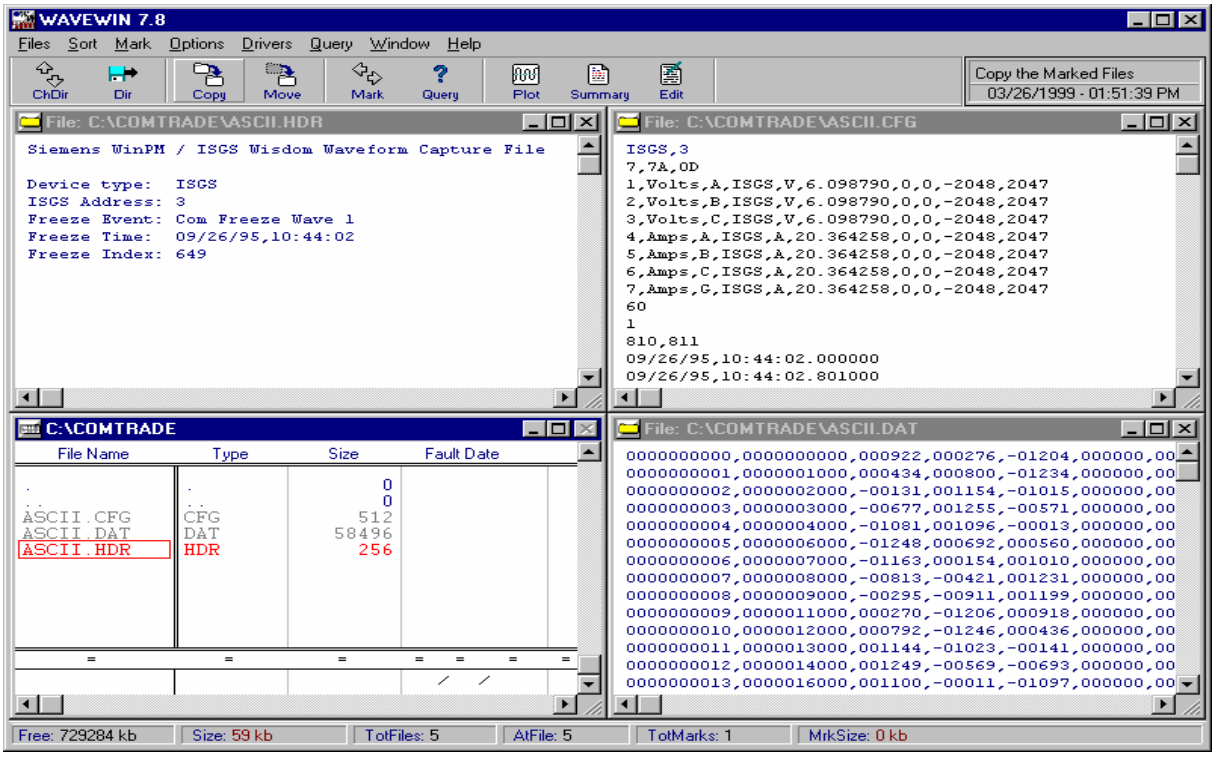


Figure-3: COMTRADE ASCII Format

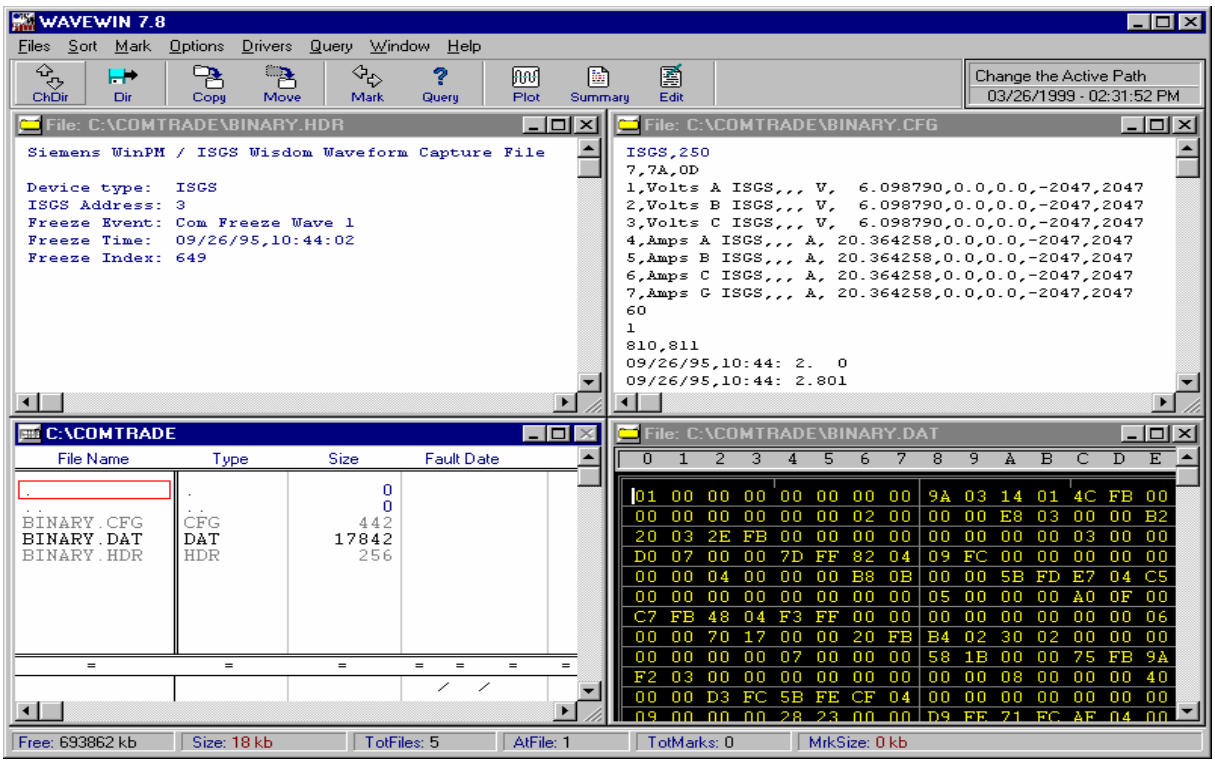


Figure-4: COMTRADE Binary Format

A fault record may be saved in one file or in multiple files. And, multiple fault records may be saved in multiple files or in one file. In other words, fault records and computer files have a many to many relationship. For example, the 1991 COMTRADE standard requires three files for each fault record: the header, configuration and data files as shown in Figure-3. The files share the same name but have different extensions: "HDR", "CFG" and "DAT": The header file contains general information about the fault record such as type, manufacturer, and operator comments. The configuration file contains specific information such as substation name, fault data and time, number of analog and digital channels, scale factors, skew values, sampling frequencies and so on. The last file is the data file and it contains the recorded data.

An example of having multiple fault records in the same file is shown in Figure-1. Such files originate from certain types of digital relays. There are also some proprietary formats that use hidden database files for the header and configuration information. Such files originate from certain types of DFR equipment as shown in Figure-2.

In order to collect, store, display and analyze fault records from a particular device, the user must use the original proprietary software that was provided for that type of device. And, almost every type of device has a different type of proprietary program. Manufacturers have provided us with too many programs and operating nuances for “data basing” fault records, which has become a hindrance to the process of fault and disturbance analysis.

Realizing this problem, manufacturers worldwide have been fast to embrace the COMTRADE standard. The vast majority of today's new devices provide their fault records in COMTRADE format allowing for the eventual rise of a common program for "data basing" fault records. The explosion of proprietary formats has been curbed and that is good news because the total number of existing formats is now a countable set. The dream of having a universal program that can handle all types of fault records is very close to being fully realized. In fact, there are a number of programs in the market today that claim universality. An example is shown in Figure-5.

DATABASES OF FAULT RECORDS:

The vast majority of fault records today are "data based" using the operating system. Fault records are saved in files and the files are named in unique ways. The names vary depending on the convictions of the manufacturer and the type of originating device. In general, today's naming conventions can be organized into five (5) classes. The classes are: associated, coded, sequenced, content addressable (IEEE-PSRC H8 report), or random (meaning user defined).

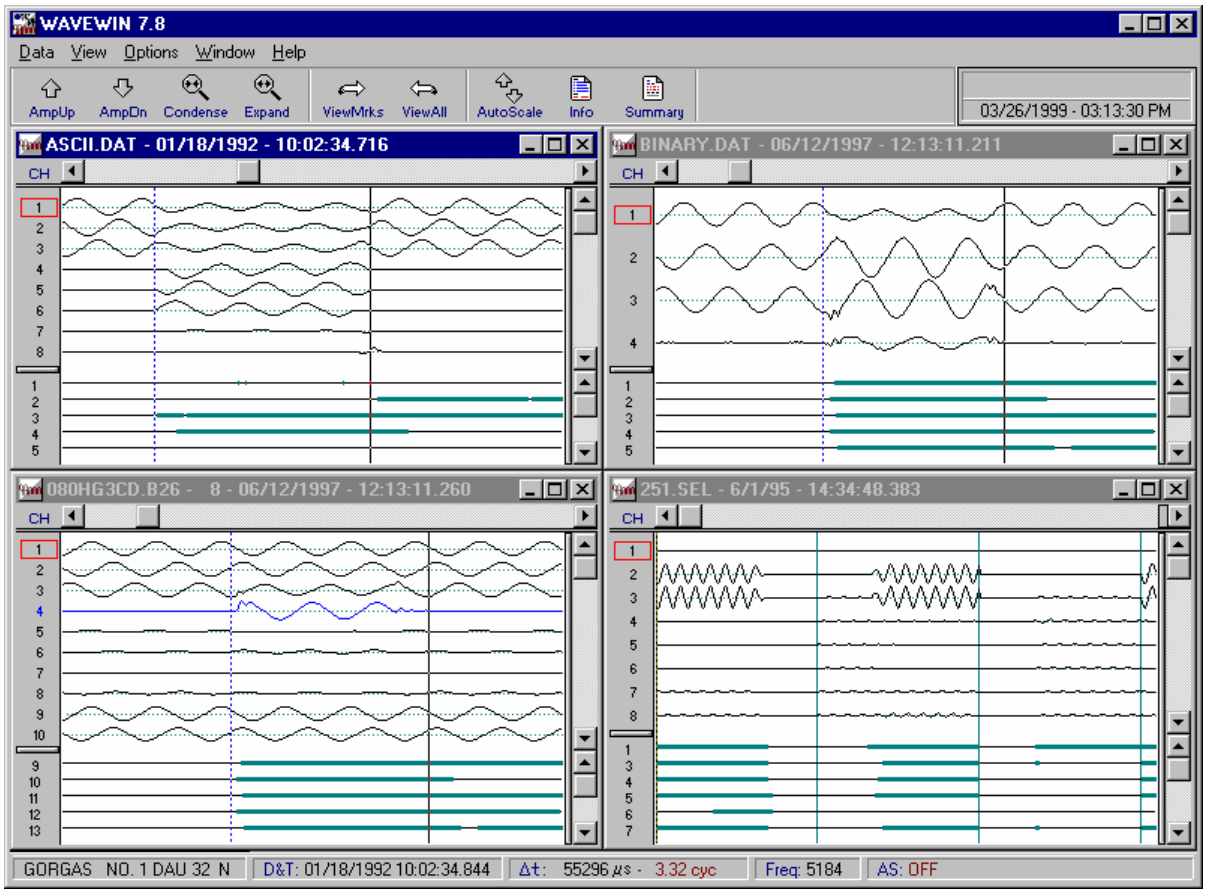


Figure-5: Universal Program For Viewing Fault Records

Associated means the filename extension defines the type of storage format. For example, the extensions "HDR", "CFG", "DAT", and "INF" are used to indicate that the file contents are compatible with the 1999 COMTRADE standard. The non-extension part of the filename, the name, is left at the discretion of the user.

Coded means the filename contains information about the fault record. The contained information is usually manufacturer specific. For example, certain DFR files have the event date and time (up to 12/31/2079-23:59:59.99) and the recorder number (up to 255) coded in the filename. The recorder number is coded in the first 2 characters of the filename and the date and time are coded in the last 9 characters. The resulting filenames are not friendly and require special decoding software. For example, “G30BQ1EF.063” is the filename assigned by recorder number 163 on 09/18/1991 at 14:15:00.630.

Sequenced means that the filenames are incrementally assigned. This method is valid because the resulting filenames are unique, however, the total number of attainable filenames is limited to the maximum value of the numerical sequence. The sequence may appear in the name or in the extension portion. When multiple devices are used then the device is also listed in the filename. For example, some filenames have the device name (4 characters) and the fault record number (4 characters = 9999 filenames before overwrite) listed in the name portion. A file named "MART1743.RCL" indicates that the event was recorded at Martin Station and the event number is 1743. The "RCL" extension means the data contents are from the first 16 analog input channels.

Content addressable means the filename contains a sufficient amount of information for users and automated analysis applications to be able to determine the extent of the file’s contents by just reading the filename. For example, the IEEE-PSRC File Naming Convention Report specifies a content addressable format that can be used to uniquely tag and identify each record within a database (or repository) of fault records. The naming convention is as follows:

“Date, Time, Code, Station, Device, Company, Optional, User. Ext”

An example filename is:

“000809,175215183,-4D,Sta80,Line717,FDC,,AG-trip,,,024miles,Zone1.REL”

There are many applications that can be realized given a unique, informative file naming system. Possible applications are: time-line manager, universal viewer, global (international) data bank of fault records and so on. Utilities, independent system operators, reliability commissions and manufacturers can now realize the benefits of sharing a common electronic filing system. Take any record from any device anywhere on the planet and just seamlessly file it.

Finally, random means that the filename assignment is up to the person who retrieves the actual fault record. In this case, it is impossible for any type of automated

application to be realized and the users will have major difficulties reporting, archiving and exchanging fault records.

FUTURE TRENDS IN DATABASES AND FAULT RECORDS:

The latest quest is to forgo the operating system as the main handler of fault records and database the records using the latest in automated development tools such as PI, Oracle, Access, and so on. That means a much deeper understanding of fault records is needed.

A DFR fault record may have up to 128 analog channels and 512 digital channels (events and sensors), and the channel designations are assigned in any order depending on the preference of the user. On the other hand, a typical fault record from a digital relay has around 8 analog channels and a few dozen digital channels, and the channel designations are assigned in order as specified by the originating manufacturer.

Analog channels are used to monitor phase (voltages and currents) and neutral current information. Up to seven (7) channels may be required in order to fully cover a line or feeder. Analog channels are also used to monitor transformer windings and polarizing potentials and may be used to measure temperature, humidity, relative saturation and so on. Digital channels are used to represent the corresponding breaker status, trigger information, trip indications, contact positions and so on.

Depending on the configuration of the equipment, the waveform data for multiple lines may be found in one fault record or the data for one line may be split among multiple records. In certain cases, the data that is needed to compose all seven (7) channels for a given line may not be available. The analog channels may be specified as just VA, VB, VC, and VN for lines where only the voltage phases are monitored, or just IA, IB, IC, and IN for lines where only the current phases are monitored.

In other cases, there may be a missing phase condition. For example the analog channels may be specified as just IA, IC, and IN. In these cases the data for the missing phase will have to be automatically calculated by subtracting the monitored phase channels from the neutral channel. In some cases, there may be more than one missing phase and the subtraction will not work.

Additionally, the fault data could be in primary or secondary values and the calibration could be in root mean square or peak-to-peak quantities. The physical parameters of the monitored lines will also have to be specified including positive and zero sequence total line impedance, and line length.

So, how should we database fault records?

A) Organize based on line name and location, or

B) Organize by channel, or

C) Organize by record, or

D) Build a sample-based system.

ORGANIZING BY LINE:

Clearly, from the above discussions, developing a line database is a very difficult task. For those developers that insist on braving the waves (authors included ☺), here is a short preview of what is needed:

First, a memory-based assembler is needed with an interpreter to access all of the various types of proprietary data formats from the DFR and digital relay devices. The interpreter must also work with the standard COMTRADE format. The interpreter has to unpack the analog and digital data from the incoming fault records and transforms them to a form suitable for storage in the database.

Next, if the data for the line's channels are successfully unpacked then the assembler can pass the newly composed line data to the database. However, if certain data is not yet available (i.e. the line is defined across multiple devices and some of these devices have not reported yet) then the assembler will have to temporarily buffer the incomplete line data in working memory.

For those lines that are defined across multiple devices, and since it is possible for the devices to be triggered at different times, the assembler should assemble only if the devices are triggered within a time frame of up to 50 milliseconds from each other. The latest trigger time should be taken as the trigger time for the assembled data and all other data channels should be time skewed by disposing of the appropriate number of pre-fault samples. The channels having the least number of post-fault samples should define the ending time, and all other channels with extra samples should be truncated.

It may also be possible that in the multiple devices case the devices were sampling at different frequencies. In that case, the lowest rate should be used as the main sampling rate and all other channels should be re-sampled accordingly (frequency matching).

The resulting line based data should be a COMTRADE like record that is centered around the fault area containing only the common time data. An example of program architecture for building a database of fault records is shown in Appendix-B.

ORGANIZING BY CHANNEL:

Alternatively, we can build a channel database. No need here to buffer, time skew, truncate cycles or frequency match. But, we still have to use an interpreter in order to

unpack the fault records and extract the channel data. The database is not that difficult to achieve. A tag can be used to describe the channel's originating device and location, plus other configuration parameters, and a pointer to the time series data stream.

ORGANIZING BY RECORD:

The simplest way is to database by record. No need here to interpret various types of waveform data. Just use the fields that are defined in the file naming convention report and point to the actual fault record in the database (as a blob file). In this way, the original raw formats of the originating manufacturers are preserved in their native form and therefore the "data based" records are considered legal fault records. The term legal means that the fault records have not been doctored or manipulated in any way by any program other than that of the originating manufacturer.

ORGANIZING BY SAMPLE:

Notwithstanding the above, building a sample based system is the "smart" thing to do. According to database theology, if each individual sample "knows all" about itself (it's origin, scale factor, sampling frequency, and so on) then the potential functionality is unlimited. This is the utopia of fault record databases. In this world, the user can launch a structured query into the database and automatically the proper samples will quickly assemble, in order, providing the equivalence of a time line manager.

EVENT BASED INTERFACES:

Regardless of how the database is organized, the user interface must be event based. Upon event occurrence the user should be notified automatically and the database should group the fault records from all of the devices that operated in the same handle. And, remember that during a storm or a major event (such as the Blackout of August 14th 2003) a very large number of devices will operate at almost the same time. The user should have, upon request, the option to browse, query, export, import, delete and so on based on the date and time of event occurrence.

ADVANCED ANALYSIS:

The main reason for having fault records is for engineers to be able to study and analyze the state and behavior of the power system. These engineers need advanced tools to help them calculate fault location or expose faulty wiring, defective devices, bad configurations, false relay operations, nasty harmonics, unbalanced circuits, overloaded assets and so on. To that extent, these tools must include a formidable array of analysis functions including but not limited to:

- Amplitude & Time Scale
- Frequency Match
- Duplicate/Truncate Cycles
- Super Impose
- Peak Detect
- RMS Measurements
- Instantaneous Measurements
- Sequence Components
- Envelope Detectors
- Frequency Filters
- Phase angles
- Harmonics
- Time Synchronization
- Event Sequences
- Virtual Channels
- Programmable Triggers

Fortunately, there are many proprietary applications that are available today and can provide such features. For example, Figure-6 shows an example of a software application having a sufficient collection of analysis tools. The main quest now is to have these programs provided as component objects that are available to the database.

ARTIFICIAL INTELLIGENCE:

Unfortunately, there are no known artificially intelligent applications today that utilize the vast wealth of available fault records. In order for the database to support such applications, it must allow for the use of some formal knowledge representation language and it must allow for the development of rules and inference mechanisms.

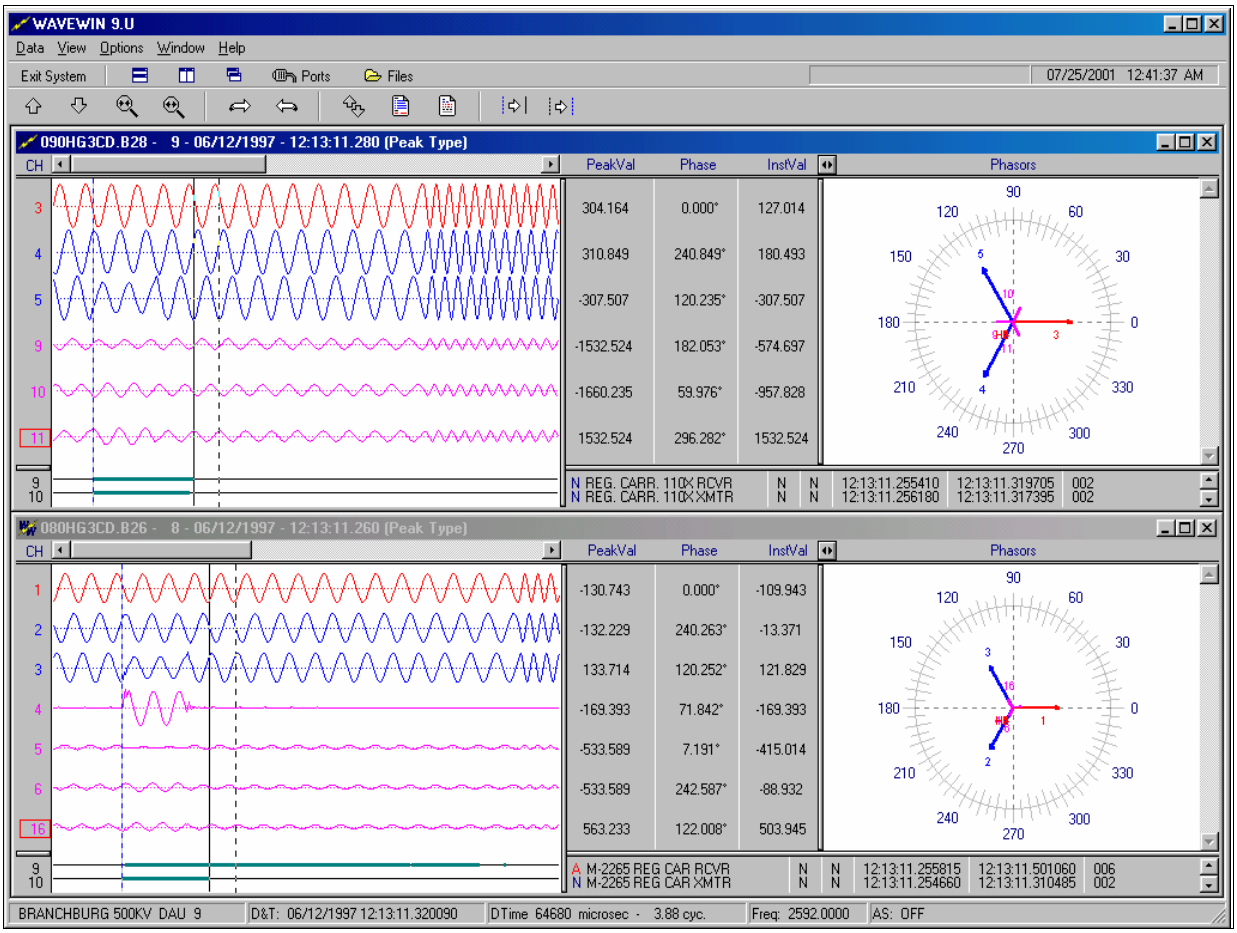


Figure- 6: Advanced Analysis Applications

CONCLUSIONS:

How do we database fault records?

We do it slowly, one step at a time. We remain patient. We keep it simple. We form working groups. We discuss, develop and refine.

Why should we database fault records?

Because, its the proper thing to do. It is always better to embrace new, proven technology than to become obsolete. Plus, the benefits of opening the information to a company (utility) are great. The information could provide Planners, Operations and Maintenance personnel, and Protection Engineers with valuable information about the power system and equipment.

The journey will take years, but the benefits will be for life.

REFERENCES:

"Transient Information Management Efficiency," A&M Makki, M. Taylor, Proceedings of the Fault and Disturbance Conference, Georgia Tech, May 2001.

"Naming Convention For Time Sequenced Data Files," H8 working group report for the IEEE Protective Systems Relaying Committee (PSRC), Draft 3.1, April 20, 2001.

"Survey of Event File Saving Schemes," A&M Makki, M. Taylor, L. Johnson Proceeding of the Fault and Disturbance Analysis Conference, Atlanta, Georgia, May 1999.

"Survey of Event File Naming Schemes," A&M Makki, M. Taylor, Proceeding of the Fault and Disturbance Analysis Conference, Atlanta, Georgia, May 1998.

"IEEE Standard Common Format for Transient Data Exchange (COMTRADE) for Power Systems," IEEE Standard P37.111, 1999.

"Overview of FAT, HPFS and NTFS File Systems," Microsoft Windows NT Workstation 4.0 Resource Kit, Chapter 18, V.1997.

"Object-Oriented Databases," Ez Nahouaraii and Fred Petry, IEEE Computer Society Press, Los Alamitos, California, 1991.

"Relational Databases," Ken S. Barathwaite, McGraw-Hill, New York, 1991.

"Distributed Database Systems," David Bell and Jane Grimson, Addison-Wesley, New York, 1992.

"Database Principles, Programming, and Performance," Patrick O'Neil and Elizabeth O'Neil, Academic Press, New York, 2001.

"Building An Object-Oriented Database System," Francois Bancilhon, Claude Delobel, and Paris Kanellakis, Morgan Kaufmann Publishers, San Mateo, CA, 1992.

"The CTDP Basic Database Guide," Version: 0.2.0, www.comtechdoc.org, June9, 2001.

APPENDIX A

DATABASE BASICS AND TERM DEFINITIONS

The term database is defined in the “Random House Webster’s Dictionary” as being: a collection of organized, related data in electronic form that can be accessed and manipulated by specialized computer software. Or, a fund of information (on one or more subjects, as a collection of articles or précis (summing-up)) that is accessible by computer.

The simplest form of a database is a directory or folder or repository of related electronic files. The Webster definition of the term directory is: a division in a hierarchical structure that organizes the storage of computer files on a disk, or a listing of such stored files. In other words, the operating system (being the handler/manager of hierarchical structures) is actually a database of files.

On the other hand, the most advanced/complex form of a database is an expert system or knowledge base. The “Webster Computer Dictionary“ definition of the term knowledge base is: a collection of knowledge expressed using a formal knowledge representation language. And the short definition of the term expert system is: a knowledge base plus a collection of rules and algorithms used to make inferences based on applied inputs.

There are several kinds of popular databases:

FLAT FILE DATABASES

Flat file databases means files of ASCII data separated by delimiters such as commas, colons, or semi-colons.

Example: 123456789, Mark, Smith, 23 N Elm Ave, Philadelphia, PA

RELATIONAL DATABASES

In relational databases, data are stored in two-dimensional structures known as tables. A table is a collection of rows (records) and columns (fields). The intersection of a row and a column is called a cell and consists of some integer, real, string or foreign key value. Foreign keys are used to point to multiple values from one cell. One or more columns can be defined as unique identifiers or primary keys. For example, in a typical transaction database, customer ID is a primary key used to identify each individual row.

Examples: Microsoft Access 2000, Oracle9i.

OBJECT ORIENTED DATABASES

In object-oriented databases, data is organized as objects. An object is any real-world entity (a person). Associated with every object is a set of attributes or fields (name, address, height, weight) and a set procedures or methods which logically operate on the object fields (calculate Body-Mass-Index). Fields and methods are stored together in the same object. This is different from relational databases where data and methods are separate. Objects that share the same methods and attributes form a "class" (an employee). Objects can also be defined as "children" of other objects and automatically "inherit" their fields and methods. Additional relationships among objects are generally established using pointers.

Examples: GemStone, Versent, Ontos.

OBJECT RELATIONAL DATABASES

These applications simply put an object oriented front end on top of a relational database. When applications interface to this type of database, it will normally interface as though the data is stored in objects. These applications can disassemble object information into tables with rows and columns and reassembled into objects upon request.

Example: Oracle9i

COMMUNICATING BETWEEN DATABASES

Databases use one or more of the following methods to exchange data.

OQL - object query language (standard language for object oriented databases).

SQL - structured query language (standard language relational databases)

API - application program interfaces of data manipulation objects.

OBJECT ORIENTED DATABASE STANDARDS

There are several object oriented standards and groups that oversee them.

Current Groups

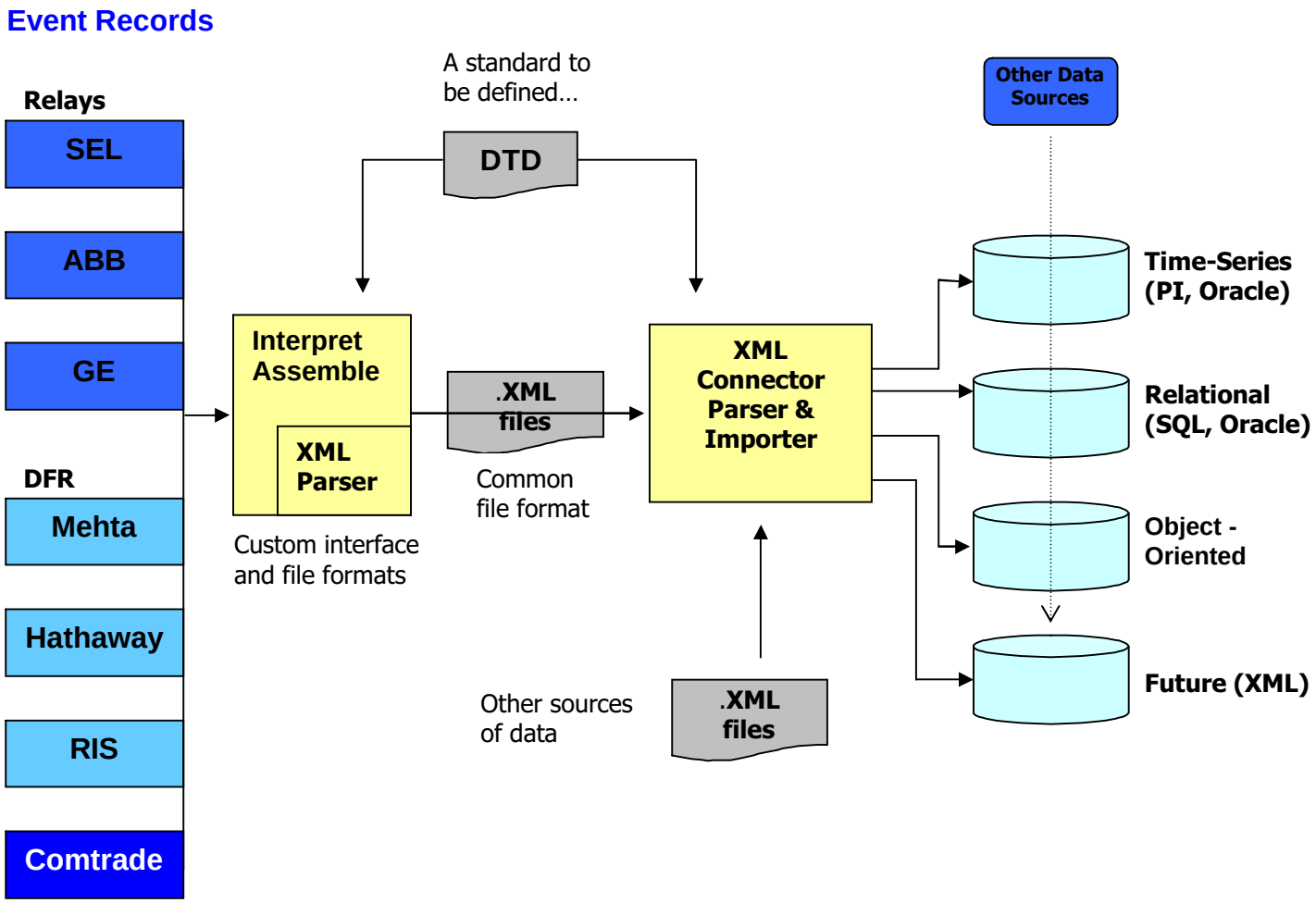
- Object Management Group (OMG) - Develops standards to help make object applications to be portable and communicate between each other (interoperability). They have developed the Component Object Request Broker Architecture (CORBA) standard along with object and OODBMS interfaces.
- Object Database Management Group (ODMG) - Created to define standard interfaces for object databases. The interfaces should allow the databases and applications that use them be portable and communicate between each other.

Current Standards

- CASE Data Interchange Format (CDIF) - Defines standards for tools that will be used from various applications such as database and application servers.
- Portable Common Tool Environment (PCTE) Standard.
- PDES/STEP - Exchange format standard for product model data. Also called the object interface format.
- Object Query Language (OQL)
- Object Definition Language (ODL) - Extension of OMG's CORBA standard.

APPENDIX B

EXAMPLE ARCHITECTURE FOR BUILDING A DATABASE OF FAULT RECORDS



The diagram above illustrates a recommended approach to transfer fault data from substation devices, primarily relays and digital event recorders, to standard database management systems where this data can be analyzed using standard tools and aggregated with other data sources (such as operations or condition monitoring data) for integrated analyses.

On the left are some various relays and event recorders that provide proprietary file formats and require specialized interfaces for extraction and analysis of fault records. The approach is to “standardize” the fault records into eXtensible Markup Language (XML) files. The format of the files will be based on the Document Type Definition (DTD) file. This file will define the structure, content and semantics of the XML fault

documents. It's this file whose definition requires a consensus of its structure. Similar to the Common Information Model (CIM) the DTD will provide a generic structure for all fault records, allowing any XML parser to "understand" the content of any fault record.

In the diagram above, the XML connector would parse the XML file as defined by the DTD and load the fault contents into a standard database format. Once the data is loaded the user may access and analyze the data via the database management systems' standard access methods and client analysis tools. The XML event files would be "understood" based on the common DTD. Once the files are in a standard or common database management system they can be integrated with other sources of operational, load management, condition monitoring, etc. data. The database management system would provide for the retention, backup, disaster recovery, etc. aspects of the data.