AuditNet
The Global Resource for Auditors

The AuditNet® Monograph Series

Principles of Computer Assisted Audit Tools and Techniques

Foreword

AuditNet® Monograph Series - Guides for Auditors

This monograph series grew out of my desire to establish an online electronic communication network for auditors. Before online services, bulletin boards and the Internet many auditors were operating without the benefits of peer collaboration and information sharing on a major scale. The Internet, founded on the principle of sharing and communication, changed the interaction model between auditors. Auditors can now post messages in online discussion forums, upload and download audit work programs, checklists, surveys, questionnaires and other audit related material in warp speed. Small one-person audit shops can now communicate with others and feel like they are not paddling upstream with one oar when it comes to having access to audit resources. My vision of an online information communication network for auditors became a reality with AuditNet® as the foundation.

The AuditNet® Monograph Series or AMS provides auditors with guidance on different aspects of the audit process and other relevant topics to help them do their jobs. New auditors will seek these guides to learn some basics of auditing while experienced auditors will use them as a review. Each guide focuses on a specific subject.

If you have an idea for additions to the AMS please send a proposal via email to editor@auditnet.org.

Jim Kaplan, AuditNet®Founder and Principal

Preface

The material included in this monograph was created by the INTOSAI Standing Committee on IT Audit for a special training program on CAATTs. INTOSAI is the professional organization of supreme audit institutions (SAIs) in countries that belong to the United Nations or its specialist agencies. SAIs play a major role in auditing government accounts and operations, and in promoting sound financial management and accountability in their governments.

AuditNet® thanks Ian Pettigrew of the UK National Audit Office for permission to reproduce this material in the form of a monograph for the benefit of the global audit community.

Table of Contents

What are CAATTs?

These notes aim to make you aware of the possibilities that CAATTs offer for carrying out a more comprehensive audit of your client's accounts and systems than would otherwise be possible within economic constraints.

1. Audit packages, such as IDEA and ACL, automate many common data manipulation routines and it is a fairly straightforward task to train auditors in their use. But understanding how your client's business systems operate, analyzing the transaction flows and then obtaining the data that you require and in a form in which you can use it, is often another matter. Depending on the scale and complexity of the processes involved, you may need to be trained in computer programming techniques and in the use of particular operating and database systems. Nevertheless we can discuss the problems that you may need to address.

2. Since organizations first used computers to maintain their accounts, auditors have recognized that the power and speed of computers can greatly assist them in their work. The term "computer assisted audit tools and techniques", or "CAATTs" for short, is used to denote techniques that you can use to help you audit in a more effective, efficient and timely manner. Whilst some CAATTs can be used for more than one purpose, there are two general categories:-

    - CAATTs for retrieving data;
    - CAATTs for verifying system controls.

3. However, before we can consider how CAATTs may be used it is first necessary to understand how computers store data. From this it will be apparent that two initial problems confront the auditor when considering the use of CAATTs. The first is to identify the data that it will be necessary to process to obtain the necessary audit information. The second problem is to obtain the data in a form that is suitable for processing. Only after these problems have been resolved satisfactorily will the auditor be able to use CAATTs.

Data storage

Files

4. If you are unfamiliar with computing, try to imagine *files, records* and *fields* as a hierarchy of relationships. We will consider each in turn.
5. A "file" is a collection of records held for a particular purpose, and it may contain more than one type of record. This is perhaps best explained by imagining a "box file" or a "card index" (see fig 1). A computer file is identical in concept. It comprises a collection of records in just the same way as a card index.

6. When a computer program reads a file (e.g. a magnetic tape or disc, or a CD-ROM), the operating system transfers a number of records from the file into the computer's random access memory (RAM) where they are made available to the computer program to process. Should the computer program need to write to an output file, the same process happens in reverse, with data being transferred as records to an output device (e.g. a magnetic or optical storage disc, a magnetic tape, a VDU, a printer, etc.).
7. Files may be organized in different ways. Again, if you are unfamiliar with computers a good analogy is to compare a music cassette tape and a music CD. With a cassette, if you want to play a particular song you need to fast forward or reverse to the position on the tape where the song begins. This can be a lengthy process. With a CD you can instruct the player to send the read head to the start of the song you want to hear, and this process takes a second or two.
8. In the world of computing a "sequential" file is one in which the records are read in sequence in much the same way as songs on a music cassette. In the case of a computer file held on magnetic tape, each record has to be read in turn until the desired record is reached. Some systems work satisfactorily on this principle. Older payroll system often used sequential files, with employee records on the payroll masterfile being sorted into pay number sequence (maybe even paynumber within employing department) and processed one after the other.
9. Disc files, however, (like a music CD) do not need to be processed sequentially (although this is an option). This is because a disc file contains an "index" into which information is entered by the operating system that allows the disc's read head to be located directly over the relevant cylinder of the disc in which the file starts. There may then be a short delay whist the sector of the disc on which the file is located rotates until it comes under the disc head, and data transfer then begins. This type of file is described as an "indexed sequential" file, and it is much quicker to process than a straight sequential file.

Records

10. A record is the main storage unit within a file. When an auditor has identified the system file that holds the data required by the CAAT, it will be necessary to obtain the file's "record layout". This describes the data that is held within each record, and the relative position of each data item. A simple record layout may look like this :

Fig 3 : RECORD LENGTH 104 characters

FIELD	**TYPE**	**START POSITION**	**LENGTH**
Account	***CHARACTER***	0	16
Record type	*CHARACTER*	16	1
Invoice	*CHARACTER*	17	13
Description	*CHARACTER*	30	50
Amount	*NUMERIC (2)*	80	8

Paid date	*CHARACTER*	88	6
Filler	*CHARACTER*	94	9

11. In the same way that a card file may contain many different types of cards (separators may separate one type from another, as in fig 1), computer files may also comprise many different types of records. For example, in a fixed assets system some types of records may relate to land and buildings, some to motor vehicles, some to computers, etc. Each record will contain information that relates to the particular asset, but it is also likely that records that relate to motor vehicles will hold different data to those that relate to land and buildings. Again the analogy to the card file holds good. Where the asset register is held on cards, those that relate to motor vehicles would probably be of a different design to those that relate to land and buildings, and would hold different information.
12. When designing CAATTs, the auditor is often confronted with this situation, a computer file that comprises different types of records. Some records in the file may be of audit interest, and others not. Some records may need to be tested in one way and others in a different manner. The auditor will need to know how to distinguish between different types of records and this information is generally shown in the record layout diagram. In the example illustrated above, you will see that the second item of data is "Record type". In this particular case 'record type' contain a simple one-character code that will allow the auditor to distinguish between different types of records and to process them accordingly.

Fields

13. A "field" is the name given to a unit of data within a record. For example, '*date of payment*' and '*amount paid*' is each a discrete item of data, or "field". In some cases a field can be broken down into other units referred to as "sub-fields". For example, a date may be expressed by its constituent parts as '*year*', '*month*' and '*day*'.
14. When you are identifying files for processing with CAATTs you must be quite clear about what each field in a record actually represents. File layouts can be vague about this. If you are unclear, then ask the appropriate application programmer or system analyst to clarify any points about how a field is used. It is preferable to guessing.

Data

Bits and bytes

15. We have now reached the bottom of the storage hierarchy, and we need to consider how data is stored at this level.
16. A computer operates on precisely timed electronic pulses. The processor reacts to pulse and no pulse patterns; storage devices hold data as 'on/off' patterns. The

binary number system is ideal for representing such patterns because it uses only two symbols, 0 and 1. We use decimal numbers because we find them convenient (probably because we have ten fingers). A computer uses binary because its structure makes binary convenient.

17. A binary digit, or "bit", is the computer's basic unit of storage. A single bit can hold either 0 or 1. Single bit fields were used in the early days of computers (as indicators or "flags") because storage space was extremely expensive and had to be used to maximum effect. Obviously there is very little that you can do with a single bit – you couldn't, for example, store a name and address in a singly bit – so bits need to be grouped together to form more useable units of storage. In practice, the smallest unit of storage that you are likely to encounter is the "byte". In the past there were computers that used different lengths of byte, but today a byte is generally accepted to be eight bits, and that is sufficient to store a single character of data.
18. Because binary numbers are so well suited to electronic devices, computers are at their most efficient when working with pure binary. A typical computer is designed around a basic unit of data called a word (PCs are either 16 or 32 bit machines). The high order bit (that on the extreme left) holds a sign (0 for +, 1 for -), whilst the remaining bits hold data. Thus the biggest value that can be stored in a 32-bit word (or 4 bytes) is…..

01111111111111111111111111111111

or in decimal, 2,147,483,647. Similarly the largest decimal number that can be stored in a 16-bit word is 32, 767.

19. Binary integers of this sort won't do when for very large, very small and fractional numbers. Scientific notation is used instead, with numbers being written as a decimal fraction followed by a power of 10. For example, the speed of light (186,000 miles per second) expressed in scientific notation is 0.186 x ten to the power of six. Many computers can store and manipulate binary approximations of scientific numbers called floating point or real numbers.

Hexadecimal

20. Life would get very tedious if the contents of fields where always expressed in binary notation – you can see from the above example that binary notation takes up a lot of space on the paper and is very confusing on the eye. In practice bits are gathered together into groups of four (known as "nibbles"), and "hexadecimal" notation is used instead. Groups of four bits can exist in any combination from 0000 to 1111 in binary, which is the equivalent of zero to 15 in decimal.
21. Finally, we need a system to identity all the possible 16 values that exist in hexadecimal (or "hex") for short. This is probably best explained by reference to the following table.

Hex Digit	Decimal Equivalent	Binary equivalent
0	0	0000

1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

23. Why represent data four bits at a time when that leads to the unfamiliar digits A to F? The answer is that hex is a reasonable compromise between what's closest to the machine (which is essentially a binary device) and what's practical for people to work with. Because the most common unit of computer data is the byte, hex can conveniently represent all the bits in a byte with two hex digits, each one representing four of the byte's eight bits. In other words two hex digits are all you need to express all the combinations in a byte. Thus 00000000 in binary is represented by the hex digits 00; 11111111 by the hex digits FF; and 01011100 by the hex digits 5C.

Text data

24. So far we have considered how data is stored within a computer, which is as binary numbers. However there are many applications that need to use data as text. These notes, for example, contain very few numbers, none of which (e.g. the paragraph numbers) are required for arithmetic purposes. They are there for display. So how do we use binary storage units (i.e. bytes) to store text? There are two parts to the answer:-

    - first we need some sort of code that enables us to represent binary numbers as text;
    - Second, we need to be able to tell the computer not to translate a text field as a number, but instead to display its contents as text.

25. Turning first to the subject of codes. There are three sets of codes in widespread use for assigning character values to numbers. EBCDIC (Extended Binary Coded Decimal Interchange Code) was originally developed by IBM and is generally used by large IBM-compatible machines. The ASCII (American Standard Code for Information Interchange) code is used more widely (including PCs), whilst Unicode is a more recent development.

26. ***EBCDIC*** is an old code that is still used by IBM compatible mainframes. It is an 8-bit code that extends to alphabetic, decimal digit and punctuation characters. A few of the EBCDIC codes are given here against their binary values (you might like to work out what the corresponding hex would be):

CODE	CHAR	CODE	CHAR
1111 0001	1	1100 0001	A
1111 0010	2	1100 0010	B
. .	. .	0101 1100	*

27. The ***ASCII*** code set is shown in full at Appendix 1. It is "*a 7-bit code defining 128 characters using unsigned integers*". These characters include control signals (the code values 0 to 31 are used for this purpose), the English alphabet (upper case and lower case letters), digits and the punctuation characters found on an (American-style) keyboard. When this code is used, the eighth bit of a byte may be used as either a *parity* bit (i.e. for error checking) or, on PCs, to provide an extra 128 graphic or alternative language codes, such as "*Latin1*" (see below).
28. The 128 characters provided in ASCII are American, not European (they don't even include a '£' character!). A number of 8-bit extensions to ASCII developed for PCs during the '80s to address these short-comings. These provided 256 characters to include European, graphics or other character sets. The *"Latin1"* character set includes characters and symbols used with many European languages that use the Roman alphabet, such as French, German and Spanish. The first 128 characters of Latin1are the standard ASCII character set, whilst the other 128 characters define the remaining European character symbols for these languages. However *Latin1*does not include symbols for the European languages that have completely different alphabetic symbols, such as Greek and Russian.
29. The ***Unicode*** international 16-bit character code provides for most of the characters used in the world languages plus others. It is designed and maintained by a non-profit making consortium. The first 128 character codes of Unicode are ASCII, and the first 256 are *Latin1*. It also:-

    - includes codes for a Han character set of about 18,000 characters for Chinese, Japanese and Korean
    - provides Arabic, Cyrillic, Hebrew, Sanskrit, Thai and other alphabet codes
    - Provides codes for Russian and Greek characters and all the mathematical symbols and the International Phonetic Alphabet. There are even the odd characters called dingbats!

30. Microsoft Windows NT uses Unicode. All the character strings used in NT, path-names, file names, directory names etc, are in Unicode. Windows NT converts any ASCII and Latin1 into Unicode before use (by adding extra empty bytes). Java, "the language of the Internet", specifies that its **char** data type is for 16-bit Unicode characters to ensure consistency across different computer systems and across continents.

31. Having dealt with the conversion codes that are necessary to convert numeric (binary) values into text, the next problem is to tell the computer when a field is text, and when it is numeric. All computer languages must have some sort of facility for doing this because there is nothing that allows the computer to distinguish between an array of bits that contains text, and one that contains an integer. Thus, when you use software, regardless of whether it is an audit package such as IDEA or a language such as COBOL, you will need to describe in your program where each field in the record lies relative to its start, and whether its contents are integer or a text. The method for doing varies from one language to another, but by way of example an FTL6 program would define the record at fig. 3 in the following way:-

```
*Dictionary
    Account   =  0/16
    Rec-type = 16/1
    Invoice  = 17/13
    Description = 30/50
    Amount  = 80:8
    Paid-date = 88/6
```

32. In this example, **Dictionary* indicates to the program compiler that the instructions that follow define the name that the programmer has assigned to each field in the record, its position relative to the start of the record, and the manner in which the field is to be used. Thus the field called *Account* starts at position zero and extents for 16 bytes. The '/' tells the compiler that Account contains text. The field called *Amount* starts at byte 80 and extends for 8 bytes, but in this case the field is to be treated as an integer – this is indicated by the use of a colon rather than a slash.
33. Programs written in COBOL and in Basic have similar facilities, although they differ in detail.

Databases

34. So far we have considered data contained in either sequential or indexed files, which are often referred to as "flat" files. In practice data is often stored in a different type of file, a "database". There are three types of database system in business use - *hierarchical, network and relational* – of which the relational database is by far the most popular.
35. Data stored within a relational database is organized and accessed according to relationships between data items. In a relational database, *files* are referred to as "tables," *records* are called "rows," and *fields* (or *data elements*) are called "columns." For example, data relating to all employees of a company might be stored in one table, the employee table. In the illustration you will see that the employee table consists of five *rows* (or records) and six *columns* (or fields).
36. More often a relational database is a collection of tables that "relate" to each other through at least one common field, such as "Department"(DEPT in the second illustration). In a relational database, for example, one's department number could

be the common thread through several data files -- *payroll*, *telephone directory, personnel,* etc, and might thus be a good way of relating all the files in one large corporate database management system.

37. Data retrieval is done through the RDBMS's "ad hoc" report capability. This gives access to all tables and data (although security constraints may restrict access to some data items to specific staff), and often represents a powerful audit tool. Appendix 1 contains a more detailed description of RDBMS's.
38. Relational databases are typically packaged and sold by software vendors as "relational database management systems" (RDBMS). Typical examples of relational databases are *Oracle*, *Sybase*, *Informix* and *Access* (Microsoft).

The applications of CAATTs

Financial auditing

Substantive testing

39. Under this approach you may, for example, gain assurance concerning the accuracy and propriety of an account by examining the constituent transactions and records. In nearly all systems it would neither be practical nor cost effective to examine large volumes of transactions and records, and certainly not 100%. Instead it is common practice to select a sample for audit examination. In some cases it may not be feasible to select such a sample without the aid of a computer, due to the volume of data to be processed and the amount of computation required. And in nearly all cases it is quicker and more efficient to use a computer.
40. The computer may thus be used to help draw a representative sample (i.e. that from which you can make the most accurate and valuable prediction concerning the whole population) as part of your substantive audit testing. In most cases the tests themselves are then applied manually once the sample has been drawn. Sampling strategies include:-

    - random sampling : where each item in the population stands an equal chance of selection;
    - interval sampling : selects every 'n' item, generally from a random start position;
    - Stratified random sampling: the population is split into strata (based upon values, quantities, etc.) and then items are selected at random from within each stratum. This allows the sample size to be adjusted according to value band, with greater emphasis being placed on high value items;
    - Monetary unit sampling: another strategy for giving greater emphasis to higher value items as they occur at random throughout the population. It involves selecting every 'n' th monetary unit (e.g. $, £, etc.).

Compliance testing

41. Certain types of controls (particularly those contained within application programs and operating system software) cannot be tested effectively using standard compliance testing procedures. For example, it may not be possible to observe a control in operation or interview the staff carrying out a check. Equally, for many such controls there is no documentary evidence that the check was carried out. There may be exception or error reports when a control failed, but no positive proof that it worked. CAATTs can provide this proof.

End-of-year tests

42. These normally consist of:-

    - both substantive and compliance tests related to end of year procedures;
    - Checks applied to verify final accounts figures (e.g. independent totaling software to reconcile individual debtor balances against a control account total).

43. With regard to the latter, you should always be careful when proving final accounts figures to ensure that your only evidence is not printouts produced by the system unless you can unreservedly be assured of their accuracy. In nearly all cases it is preferable to use independent audit software to provide reconciliation between account balances and the underlying transactions.

Analytical review and predictive analysis

44. This is a technique by which comparable balances are compared between accounting periods, or from figure to figure (e.g. by using accounting ratios) after taking into account the movements in major determining factors. In simple cases such techniques can be applied manually, but in more complex situations where a number of interrelated but differently weighted factors are involved it is more efficient to use CAATTs.

Efficiency analysis

45. This is a specialized branch of computer auditing that is included here for completeness.
46. Efficiency analysis involves an examination of the use that is being made of a major asset, the client's computers. The aim is to ensure that an adequate throughput of work is being achieved for the minimum equipment expenditure, having due regard to varying workloads and the need for resilient systems. This involves an analysis of central processor and peripheral use, and most operating systems are equipped with software tools to enable this type of work to be carried out.
47. It may also be necessary to assess the effectiveness of resource allocation, and the possible charging of computer facilities across jobs to specific departments and users.

VFM studies

48. This is a very wide field in that there are many opportunities to use CAATTs. For example, the selection of particular types of records (e.g. identification of slow moving stores items, or stores items past their shelf life; debtors over 2 years old) or the analysis of different component parts of an expenditure figure and relative trends over recent years. The examination of such records and trends may clearly highlight areas where value for money appears at risk or where an organization is being wasteful or extravagant.

CAATTs - Techniques

Introduction

49. Before discussing the various types of CAATTs in more detail, there are several important points that apply to the application of all audit software.
50. Much time is wasted by auditors who don't fully understand how the target system works, and how its files/databases are constructed, and they make invalid assumptions to fill deficiencies in their knowledge. If the auditor is lucky the audit software throws up results that are obviously wrong, but sometimes the results – whilst equally worthless – appear feasible and time can be wasted before this becomes apparent. This takes us on to the second general rule, record your work.
51. Document your audit objectives; the system under audit; the relevant file specifications and relevant record types and how to recognize them. Insert notes throughout the body of your program to describe what it does at that particular stage. Your manager will need to review your work, and other auditors may need to run your audit software on the same system at a later date. Record the names of key contacts in the client's IT department who can answer questions on system operation. Their help is often vital.
52. Test your work to prove that it works. In this respect you are just another development programmer, and you should employ the same methods that you would expect a program development team to adopt. If possible, get someone else to review and test it, because no matter how hard you try you can never achieve an objective view of your own work.
53. When you are running CAATTs, make sure that you create control records that you can reconcile with the client's ledgers to confirm that you have processed all the data from which the audited accounts are drawn.
54. Don't expect the target system to remain static for very long. IT systems are changed frequently to cater for enhancements, and changes often cause file and database structures to be altered. Before you repeat a CAAT that you ran successfully on an earlier occasion, confirm with the client's application programming team that there have been no system changes in the intervening period that have altered system operation, and/or the structure of your target file or database.

55. Never connect your PC to your client's system without their written permission. And if you use your client's on-line system, make quite sure that you can only access the data in "read only" mode.
56. Finally, make sure that the CAATTs exercise is going to be cost-effective. It can be an expensive and time-consuming process to analysis a system, identify the target file(s), and perform any preliminary processing and data conversion that may be necessary, and then develop and test your CAAT. You may only use the CAAT once, or if the system changes frequently – as some do – you may need to update your software before you can use it again. There are occasions when the most cost-effective method is to use the client's on-line enquiry system, if one exists, or obtain a hard copy print of all the transactions on the file!

CAATTs : Objectives

57. CAATTs fall into two general categories. The first involves examinations of computerized data. Within this context, data files may hold either transaction data or standing data files. CAATTs are not confined to accounting data alone, but may be used for processing non-accounting files such as journals (or "logs") that are created when accounting data is processed. Two types of CAATTs are commonly used for reviewing file data. They are:-

    - data file interrogation;
    - Embedded audit modules.

58. By contrast with techniques that review file data, CAATTs in this category are designed to test controls within the system. You are then able to judge how reliable the controls are and how accurate the accounting and other records may be. Techniques (described in more detail later) that are commonly used to review and verify system controls include:-

    - test data;
    - integrated test facilities (ITF);
    - parallel simulation;
    - program code comparison;
    - Program code review.

59. We will now consider each of these types of techniques in more detail.

Data File Interrogation

60. Data file interrogation is about using audit software to review information held in computer files, and to use the computer's speed and reliability helps you to cope with the massive volumes of data often involved. The sorts of operations that auditors often need to perform on data include:-

    - selecting records that conform to particular criteria;

- printing selected records for detailed examination;
- printing totals and subtotals from an accounting file;
- reporting on file contents by value bands (“stratification”);
- searching for duplicate transactions;
- searching for gaps in sequences;
- comparing the contents of two (or sometimes more!) files, and printing either record matches (where none should match) or exceptions (where all should match);
- Sorting and merging files in preparation for other audit tests (such as file comparisons and gap analysis).

61. A further type of data file interrogation is in the application of various types of sampling techniques, including:-

    - random sampling
    - interval sampling
    - monetary unit sampling
    - Cell monetary unit sampling.

Embedded audit modules

62. An embedded audit module is a technique that is generally used with a computer system that handles very high volumes of data. As its name implies, it’s an audit application that is permanently resident within the main processing system.
63. The embedded audit module examines each transaction as it enters the system. Every time a transaction occurs that meets the selection criteria, transaction details are logged before the transaction is allowed to continue for further processing. The audit log file is periodically scanned, analyzed and reports are printed for follow up.
64. Embedded audit facilities usually have the ability to select transactions that fulfill a range of criteria, which may be altered by amending the selection parameters.
65. An embedded audit module is not something that can easily be added to a system once it is operational. Its design needs to be considered carefully to ensure that it intercepts transactions at the most appropriate stage of processing, that the operation of the module does not degrade system performance, and that the audit selection parameters and log files are protected against unauthorized alteration. If you decide on an embedded audit module, you will need to specify your requirements before work is started on system design, and provide cost justification for the added complication.
66. There are two approaches to this technique. That illustrated, and described above, is referred to as “embedded data collection”. The other approach is known as “tagging”. Here selected records are merely tagged; an extra field is added to each selected record to enable it to be identified easily for audit purposes at a later stage in the process.

Test data

67. There may be occasions when you might wish to verify the correct operation of particular program or module, and this may be done in several ways. It may be possible to reconcile the input data with the output that it gives rise to, and by this means confirm that the software is working as intended. Alternatively, you might process some data yourself to exercise the program logic, in exactly the same way as the development programmer does when testing a new or amended piece of software for design and coding errors. This is the principle underlying the use of "test data".
68. Test data is generally used to confirm the operation of new or amended programs, or programs that generate output that cannot easily be predicted or reconciled with input. It can be used to test and verify:-

    - input validation routines;
    - error detection capabilities;
    - processing logic and calculations;
    - the accuracy of reports;
    - Any manual procedures surrounding the system, although one needs to ensure that the test data is submitted at a point in the process where manual routines apply.

69. On the face of it test data might seem to be an ideal solution to the problem of checking program operation. The obvious advantages offered by this technique are that it:-

    - requires limited technical knowledge;
    - is usually fairly simple to operate;
    - Helps the auditor learn how the system operates.

70. However, there are also some significant drawbacks, not least of which is that in normal operation, software is often changed to introduce new facilities and to correct bugs, and the assurance gained from your tests may be short-lived. In general, you should be aware that:-

    - Test data only confirms the operation of the program at the time that it is tested. It subsequently be amended;
    - If the program is tested on a development computer, you gain no assurance that this is exactly the same software that is operating on the live machine, and that all the program interfaces on the test computer are also identical to those on the live machine. You also need to confirm that the client has effective "configuration management" and "change control" procedures. These help ensure that software that has successfully passed quality control following its development or procurement is securely protected from unauthorized amendment. This will provide assurance that

the software is a true copy of that in live use, and that it will not be subject to unauthorized amendment following testing.

- You may not be testing the program with live data, and even if you are there is no guarantee that it contains all the combinations of circumstances that may arise in live operation. In live operation unpredictable combinations of circumstances sometimes arise that result in problems because they were not anticipated during program design. You need to pay a great deal of attention to how you design test data to ensure that you exercise as many program functions as possible. It is important to design test data that confirms not only that the program will do what it is meant to do, but also that it will not do what it is not meant to do.

71. Please do not be put off by these warnings; be aware of them.

Integrated Test Facility

72. An Integrated Test Facility ("ITF") is a technique that is sometimes used in auditing complex application systems. It provides an in-built testing facility through the creation of a dummy department or branch within the normal accounting system. Banks sometimes create such a test branch within their Customer Accounting System that is used both for audit testing and for training tellers in the use of the terminal system.
73. Given that a test or audit branch has been created on the system, the audit methods are much the same as for test data. The bank's auditor will be provided with his own audit terminal. He may set up new customers on the master file, and he may input test deposits, withdrawals, standing orders and other transactions. In all respects he can operate the audit branch as any normal branch. The transactions would be processed as normal, and would be reported by the normal branch reporting system.
74. The vital difference is that a small number of significant checks need to be written into the program to prevent the auditor's test data corrupting the live system. For instance, audit transactions must not be:-

    - incorporated into the bank's management accounting data;
    - allowed to corrupt government banking statistics;
    - Allowed to generate funds transfer to or from live accounts!

75. The advantages of using an ITF are:-

    - it allows regular comprehensive testing of the live system;
    - testing can be unscheduled and unknown to other staff;
    - small operational costs are involved once it is set up;
    - it provides prima facie evidence of correct program functions;
    - It can be used for system testing, user training, etc.

76. Again, there is a need to specify requirements early in the system development process so that the impact of the ITF on the overall system can be considered, and it can be integrated into the system design.

Parallel Simulation

77. The objective of parallel simulation is to generate an independent program to simulate part of an application. For example, suppose that you want to prove that an interest calculation program works properly, but because of excessively high data volumes you are unable to do this easily. You may decide that to test real data you should write your own interest calculation program. If you run your simulation program against the same source data file that is submitted to the operational interest calculation suite, you should obtain the same results.
78. Such a test is not always as daunting as it may seem at first sight. Since the simulation program will only be concerned with one or two aspects of the operational program, it will usually be smaller and much less complex.
79. As with test data you must remember that the results that you obtain are only relate to the test that you performed. The program may be amended later, and your tests may need to be repeated.

Program Code Comparison

80. Utility programs are available that will compare two versions of a program, and report difference between the two. This approach is sometimes used by configuration managers to compare programs returned after amendment with the previous version held in the definitive program libraries. All variations between the definitive and the amended modules are identified, and can then be checked to ensure that only authorized changes have been made. Configuration Managers sometimes carry out similar tests, but between definitive versions and those in live use. This activity is referred to as “configuration audit”.

Program Code Review

81. Not really CAATTs, but sometimes regarded as such. Code review involves a detailed examination of program coding. It generally involves a fair degree of programming skill, and a thorough knowledge of program specification. As with some of the other techniques, it provides no guarantee that the code under review is actually that in live use unless the client has sound change control and configuration management procedures in place.

Steps in using CAATTs

Overview

82. To use any type of CAAT you should first understand what it is you aim to achieve, and why. You must then gain a thorough understanding of the way in

which the target system operates in order that you can identify the file(s) that contain the data that you require and their structure. You will also need to understand the structure of the records they contain since you will need to describe these in your interrogation program. A lot of time can be wasted in performing interrogations that fail due to guesswork and incorrect assumptions; it is far better to invest time to research system operation and obtain the facts.

83. You must next determine the criteria for selecting the records you require (assuming that you will not require all record types, which is often the case), and the extent of any sampling routine you intend to carry out. Similarly you will need to work out whether any calculations are required – at the very least you will need to produce some totals that you can reconcile with your client's ledgers to confirm that you have processed all the data from which the audited accounts were prepared. You may need to perform more complex calculations, which require you to read several different files. For example when checking amounts of depreciation on fixed assets, you may require one file containing the asset records to be depreciated, and another containing the depreciation amounts that have been posted into the accounts.
84. When you have decided what interrogations you need to perform, you will need to determine at what point in the processing cycle your interrogation should be conducted, and if necessary arrange for copies of the live files to be taken for your use. You need to take care that you do in fact interrogate the correct version of the file in question; otherwise you run the risk of obtaining misleading results.
85. Finally you will need to consider how best to present the output from your interrogations. It is wise to write your output to a magnetic file together with a summary report before you print any details.

Specifying the format of the data file

86. When the most appropriate file has been selected for CAATTs interrogation and the information required from that file has been specified, it is necessary to agree a format for the data and a method of its storage.
87. The data should be provided in a format that can be readily accessed by the auditor's PC, and on media that can be read by the hardware used by the auditor. This can be a problem with data supplied on magnetic tape, because of the variety of tape formats in use.
88. The data needs to be in the simplest form possible preferably a form that can be linked directly to the CAATTs software (e.g. IDEA) being used. If the audit file cannot be provided in a readily usable form it may be necessary to manipulate the data prior to interrogation. If this is the case the data should be provided in a format specified by the CAATTs specialist.
89. If the client department is unable to provide data in a format readily usable by the audit department, it is advisable to ask for a report file. All computer systems are able to output some form of printed report, which could be directed to a magnetic file (e.g. a floppy disc) instead of a printer. Many utilities exist that can manipulate report files into auditable tables. However, you should avoid system backups, which are often very difficult to deal with.

The Standard Requirements for data

90. These are the standard requests for data format and storage media as used by the National Audit Office. Requirements will differ from client to client because the hardware and software used generally differ:-

- The data is fixed length text format. Variable length records are more difficult to link to file conversion software;

- The data is provided in a format compatible with the software used by the auditor (Windows files such as Lotus or Excel, Dbase, Comma Separated ASCII.....);
- if a client is unable to produce data in one of the required formats it is always possible to print the data to a floppy disk, instead of a printer, and use a report file utility to create a usable file;
- The data must be provided on a magnetic source that is compatible with the hardware used by the auditor. It may be possible to download data using a modem, or a PC to PC utility such as Laplink. If this is an option, it is advisable to discuss the implications of virus contamination (macro viruses) with the IT security staff from both the client department and the audit department;
- In some instances it may be possible to set up a terminal in the audit department to access the client's system. Data can then be extracted directly from the client body to the auditor's PC;
- It is vital that the audited body provides a record layout with the data to be interrogated showing the fields used and the data types that exist. This information helps the auditor when downloading and converting data, and linking it to the interrogation software.

CAATTs - Software

91. Programs used by auditors to interrogate files, generally known as "audit software", come in a number of forms ranging from packages that have been specially designed to support auditing, to any other software that the auditor finds useful. Audit software falls into the following groups. However, it should not be forgotten that general-purpose business tools, such as spreadsheets and databases can perform a wide range of basic financial and statistical functions. They usually include their own internal programming languages.

Data retrieval packages

92. There are a number of audit packages on the market; those in common use include IDEA, ACL, EASITRIEVE and PANAUDIT (see Appendix 1).
93. IDEA is used extensively by the National Audit Office. It includes a full range of auditing tools in a standalone package that can be installed on a PC under Windows 95. It includes extensive help facilities. The types of facilities it offers include:-

- sorting and indexing files;
- extracting records from a file based on selection criteria;
- statistical analysis;
- detection of duplicate records;
- file comparison;
- Stratified sampling.

94. All the above activities can be carried out with minimum programming knowledge, using a series of structured menu options and on-line help.

95. The main problem in using microcomputer-based packages is that they must first receive their data from the host system before it can be processed. This may represent a problem if the data is stored in an unusual format, or is in such large volumes that it exceeds the capacity of the local PC disc drive. IDEA attempts to get around the first problem by providing a range of data conversion utilities, which can translate to and from most of the common spreadsheet and database formats in use. They are also able to translate between the ASCII data format used by most computer systems, and the EDBCDIC data format favored by IBM for its mainframe range.
96. The second problem, which is that of data storage, is becoming less serious with the development of high capacity disc drives. At the time of writing it is not unusual to find even entry level PCs being offered fitted with 15GB discs. However, data volume can still remain a problem in large government systems. The usual technique is for the auditor to carry out some preliminary interrogation on the host machine to extract only those data items that are required to support the audit. This generally reduces the volume of data to a manageable level. Preliminary interrogation of this sort also provides the auditor with an opportunity to simplify record structure, should this be necessary, to one that will be easier to import into a general purpose audit package (e.g. IDEA).

Manufacturers' utility software

97. Nearly all computer manufacturers supply utility programs, often as additions to the computer operating system. Although these programs may not be specifically designed for auditors, they are often useful for performing data "pre-processing" (i.e. manipulating the data into a form in which it may be used by an audit package). These operations may include extracting specific data items from a database; sorting, merging or joining complete files, or specific records within them; and taking hexadecimal record prints to examine file contents more closely.

Enquiry programs

98. There are many packaged financial accounting systems on the market. Indeed, it is now very unusual within UK government to find a department developing its own financial accounting systems rather than use an off-the-shelf package such as ORACLE, Sun Systems, SAP, etc. These products generally come with their own

general enquiry packages built in, and these can be used for selecting items that are of interest for downloading to a PC.

Programming languages

99. Utility programs, enquiry programs and audit packages automate many common programming functions (opening, closing, sorting and comparing files; automatic totaling and stratification, etc). However auditors sometimes need to construct audit software using a programming language when the complexity of data processing is beyond what a data retrieval package will support. Such purpose-built audit software tends to be highly specific in what it will do, can be time-consuming to produce and, unless properly specified, can also be inflexible in operation.
100. **It is also essential that you test your audit software thoroughly**, otherwise the results produced could be misleading without you realizing it.
101. Any language can be used for audit purposes including standards such as 'C', COBOL, BASIC and PL/1. However at this level of CAATTs, the auditor will probably need to be trained in the language concerned, in program design, and in the use of the software within a particular operating system environment, such as Unix. At this level CAATTs really becomes a subject for the specialist.

Appendix 1: the 128 ASCII character codes given as decimal values

CODE	CHAR	CODE	CHAR	CODE	CHAR	CODE	CHAR
0	Null char	32	(space)	64	@	96	'
1	Start of Heading	33	!	65	A	97	a
2	Start of Text	34	"	66	B	98	b
3	End of Text	35	#	67	C	99	c
4	End of Transmission	36	$	68	D	100	d
5	Enquiry	37	%	69	E	101	e
6	Acknowledge	38	&	70	F	102	f
7	Ring Bell	39	'	71	G	103	g
8	Backspace	40	(	72	H	104	h
9	Horizontal Tab	41	)	73	I	105	i
10	Line Feed	42	*	74	J	106	j
11	Vertical Tab	43	+	75	K	107	k
12	Form Feed	44	,	76	L	108	l
13	Carriage Return	45	-	77	M	109	m
14	Shift Out	46	.	78	N	110	n
15	Shift In	47	/	79	O	111	o
16	Data Link Escape	48	0	80	P	112	p
17	Device Control 1	49	1	81	Q	113	q

18	Device Control 2	50	2	82	R	114	r
19	Device Control 3	51	3	83	S	115	s
20	Device Control 4	52	4	84	T	116	t
21	Negative Acknowledge	53	5	85	U	117	u
22	Synchronise Idle	54	6	86	V	118	v
23	End of Trans Block	55	7	87	W	119	w
24	Cancel	56	8	88	X	120	x
25	End of Medium	57	9	89	Y	121	y
26	Substitute	58	:	90	Z	122	z
27	Escape	59	;	91	[	123	{
28	File Separator	60	<	92	\	124	|
29	Group Separator	61	=	93	]	125	}
30	Record Separator	62	>	94	^	126	~
31	Unit Separator	63	?	95	_	127	Delete

Appendix 2: The Relational Database

Introduction

The relational database model has become the de-facto standard for the design of databases both large and small. While the concepts involved are not terribly complex, it can be difficult at first to gain an understanding. These notes cover the following topics:

- Primary and Foreign Keys
- Queries
- Structured Query Language (SQL)
- Referential Integrity
- Normalization

The simplest model for a database is a flat file. You have only a single table, which includes fields for each element you need to store. Nearly everyone has worked with flat file databases, at least in the form of spreadsheets. The problem with flat files is that they waste storage space and can be difficult to maintain. For example, consider a customer order entry system. Assume that you're managing the data for a company with a number of customers, each of which will be placing multiple orders. In addition, each order can have one or more items.
Before moving on, let’s describe the data that we wish to record for each component of the application:

- Customers
- Customer Number
- Company Name
- Address
- City, State, ZIP Code
- Phone Number
- Orders

- Order Number
- Order Date
- PO Number
- Order Line Items
- Item Number
- Description
- Quantity
- Price

It doesn't take a database design expert to see what the problem is in using a flat file to represent this data. Each time an order is placed, you'll need to repeat the customer information, including the Customer Number, Company Name, etc. What's worse is that for each item, you not only need to repeat the order information such as the Order Number and Order Date, but you also need to continue repeating the customer information as well. Let’s say there's one customer who has placed two orders, each with four line items. To maintain this tiny amount of information, you need to enter the Customer Number and Company Name eight times. If the company should send you a change of address, the number of records you need to update is equal to the sum of product of orders and order line items. Obviously this will quickly become unacceptable in terms of both the effort required to maintain the data and the likelihood that at some point there will be data entry errors and the customer address will be inconsistent between records.
The solution to this problem is to use a relational model for the data. This simply means that in this example each order entered is related to a customer record, and each line item is related to an order record. A relational database management system (RDBMS) is then a piece of software that manages groups of records that are related to one another. Let's take our flat file and break it up into three tables: Customers, Orders, and OrderDetails. The fields are just as they are shown above, with a few additions. To the Orders table, we will add a Customer Number field, and to the OrderDetails table we will add an Order Number field. Here's the list again with the required additional fields and modified field names.

- Customers
    - CustID
    - CustName
    - CustAddress
    - CustCity
    - CustState
    - CustZIP
    - CustPhone

Orders

    - OrdID
    - OrdCustID

    - OrdDate
    - OrdPONumber

- OrderDetails

    - ODID
    - ODOrdID
    - ODDescription
    - ODQty
    - ODPrice

What we've done besides the name change (each field name is now prefixed by its table name) is to add fields to the Orders and OrderDetails tables. Each has key fields used to provide a link to the associated Customers and Orders records, respectively. These additional fields are called foreign keys.

Primary and Foreign Keys

There are two types of key fields we are dealing with: primary keys and foreign keys. A primary key is a field that uniquely identifies a record in a table. No two records can have the same value for a primary key. Each value in a primary key will identify one and only one record. A foreign key represents the value of primary key for a related table. Foreign keys are the cornerstones of relational databases. In the Orders table, the OrdCustID field would hold the value of the CustID field for the customer who placed the order. By doing this, we can attach the information for the customer record to the order by storing only the one value. We'll discuss how to put the data back together again next.

Queries

So far we've broken down our order entry system into three tables and added foreign keys to the Orders and OrderDetails tables. Now, rather than repeating the Customers table data for each Orders table record, we simply record a customer number in the OrdCustID field. By doing this, we can change the information in the Customers table record and have that change be reflected in every order placed by the customer. This is accomplished by using queries to reassemble the data.
One of the inherent problems of any type of data management system is that ultimately the human users of the system will only be able to view data in two dimensions, which in the end become rows and columns in a table either on the screen or on paper. While people can conceptualize objects in three dimensions, it’s very difficult to represent detail data in anything other than a flat table. After all the effort we went through to break down the original flat file into three tables, we are now going to undo that effort and make a flat file again.
We're going to accomplish this amazing feat of backward progress by using queries. A query is simply a view of data that represents the data from one or more tables. Let’s say we want to see the orders placed by our customers. We can link the Customers and Orders tables using the CustID field from Customers and the OrdCustID field from

Orders - remember the value of the OrdCustID field represents a related record in the Customers table and is equal to the CustID value from that record. By joining together the two tables based on this relationship, we can add fields from both tables and see all orders along with any pertinent customer data.

Structured Query Language

Queries are built in a relational database using Structured Query Language, or “SQL”. SQL is the standard language for relational databases and includes the capability of manipulating both the structure of a database and its data. In its most common form, SQL is used to create a simple SELECT query. Without getting into all the details now, suffice it to say that if you will be doing any serious work with databases, you're going to need to learn SQL.
Let's take the earlier example and build a query to look at customer orders. Here's the SQL for it:

```
SELECT CustName, CustCity, CustState, OrdDate
FROM Customers INNER JOIN Orders ON
Customer.CustID = Orders.OrdCustID;
```

This query starts with the SELECT keyword, and most of the queries are SELECT queries. SELECT simply means that we wish to "select" records, or retrieve records from the tables. Following the SELECT keyword is the list of fields. Next comes the FROM keyword. This is used to indicate where the data is coming from. In this case, it’s coming from the Customers table and the Orders table. The key to this query is the INNER JOIN. There are two basic types of joins that can be done between tables: inner joins and outer joins. An inner join will return records for which only the matching fields in both tables are equal. An outer join will return all the records from one table, and only the matching records from the other table. Outer joins are further divided into left joins and right joins. The left or right specifies which side of the join returns all records. The balance of the example query specifies which fields are used to join the table. In this case we are matching the CustID field from Customers to the OrdCustID field (the foreign key) in Orders.
Each RDBMS has its own particular dialect of SQL.

Referential Integrity

Let’s consider what happens when you start manipulating the records involved in the order entry system. You can edit the customer information at will without any ill effects, but what would happen if you needed to delete a customer? If the customer has orders, the orders will be orphaned. Clearly you can't have an order placed by a non-existent customer, so you must have a means in place to enforce that for each order, there is a corresponding customer. This is the basis of enforcing referential integrity. There are two ways that you can enforce the validity of the data in this situation. One is by cascading deletions through the related tables; the other is by preventing deletions when related records exist.
Database applications have several choices available for enforcing referential integrity, but if possible, you should let the database engine do its job and handle this for you. The

latest advanced database engines allow you to use declarative referential integrity. You specify a relationship between tables at design time, indicating if updates and deletes will cascade through related tables. If cascading updates are enabled, changes to the primary key in a table are propagated through related tables. If cascading deletes are enabled, deletions from a table are propagated through related tables.
Looking again at our order entry system, if cascading updates are enabled, a change to the CustID for a Customers table record would change all of the related OrdCustID values in the Orders table. If cascading deletes are enabled, deleting a record from Customers would delete any related records in the Orders table. In contrast, if cascading updates or deletes are not enabled, you would be prevented from changing the primary key or deleting a record from Customers if any related records exist in the Orders table.
Keep in mind also that if you have enforced referential integrity in the relationship between Orders and OrderDetails, this relationship can also have an effect on your ability to manage records in Customers. Just as you can't delete a customer with orders, neither can you delete an order with detail items. The result is passed along as far as necessary. If you cascade deletes from Customers to Orders, but not from Orders to OrderDetails, you will be prevented from deleting a record in Customers if there are any Orders records which have related OrderDetails records.
Before you go ahead and enable cascading deletes on all your relationships keep in mind that this can be a dangerous practice in some situations. Let’s say you have a table called States, which lists two letter state abbreviations for each of the states in the country, along with the full name of the state. You use this table as a lookup table and to enforce the validity of the state entered in the Customers table. If you define a relationship between the States table and the Customers table with cascading deletes enabled, then delete a record from States, you will delete all Customers table records where the customer is located in that state.

Normalization

Normalization is simply the process of distilling the structure of the database to the point where you have removed repeating groups of data into separate tables. In our example, we have normalized customers and orders by creating a separate table for the orders.
If you look at the Customers table, you can see that it isn't really necessary to include the CustCity and CustState fields since a zipcode uniquely defines a city. However, when taken to extremes, you'll pay a performance penalty for excessive normalization. If you were to fully normalize the Customers table, you would need to remove the CustCity and CustState fields and create a table, perhaps called ZipCodes, which included these fields. Then include only the CustZIP field and join the Customers table to the ZIPCodes table in order to reconstruct the full address. The problem with this is that you add the overhead of an additional join in every query where you need to have the full address available.
There aren't any hard and fast rules for when to stop normalizing a database. You need to make your own choices based on the practicality of the data structures and the performance trade-offs involved. If possible, you should at least try to design the application so that you can restructure the data to accommodate normalizing or demoralizing the tables.

Summary

Although that's a lot of information to absorb, it’s only the "tip of the iceberg" in database design. The key concepts that you must understand in order to design a database properly are primary and foreign keys, which are used to define relationships, referential integrity, which is used to maintain the validity of the data, and normalization, which is used to develop a data structure. Once you have these concepts down, the rest of the details will fall into place more easily.

Appendix 3: Data Analysis Tools

Idea5

Idea5 is one of the leading file interrogation packages available at the moment. This software is a very powerful interrogation tool. Its features include:

- Sampling - Cell MUS, Random, Systematic, Attribute, Stratified
- Totaling
- Stratification
- Indexing/Sorting
- File Comparison
- Data Extraction
- Using Idea5 it is possible to perform all of the interrogations required for financial and VFM audit;
- It is possible to import files of many formats into Idea5. On occasion it may be necessary to perform some form of file conversion prior to importing data (VME) although this is rare;
- Idea5 is capable of exporting data into most of the commonly used formats such as Lotus, Dbase, Comma separated, and Fixed length text, producing windows compatible files;
- It is possible to use Idea5 as a file-downloading tool due to its ability to access a tape drive directly. The multi format import and export facilities enable the user to import a file in one format and export that file into another format compatible with the audit department’s file interrogation software;

Idea for Windows

The Windows version of Idea has recently been released. This is a fully Windows based package possessing all of the common windows functionality.
Whilst not being able to link directly to a tape Drive, the package possesses the cut and paste features common to all windows packages. These include OLE and ODBC (On-line Database Connectivity) allowing the user to link to a range of files (ACCESS, EXCEL etc.) directly using the ODBC drivers supplied with Windows.
This version of Idea has all of the Windows functionality:

- Cut and Paste
- Multiple Windows
- Intelligent Toolbars
- Windows Help

Go to Caseware-IDEA for the latest version of their software

ACL

The ACL file interrogation software is the main rival to Idea5 and Idea for Windows in the field of file interrogation. This software possesses all of the functionality of Idea5 for DOS
Go to ACL's Web site for the latest version of their software.

Applaud

Applaud is the third most popular file interrogation tool on the market. It is able to perform most of the interrogation functions of both Idea and ACL. The software is however more "memory hungry" requiring a minimum of 500k to run properly. The software does not have such a user-friendly interface and is unable to import data from well-known packages directly.

Go to Premier Web site for the latest version of their software.

Prospector

Prospector falls in between file downloading and file interrogation tools. The software is able to interface with a 9-track half-inch magnetic tape reader thus eliminating the need to download all of the audit data to hard disk. The software recognizes all of the common data types and formats enabling a multi for import and export. This tool is primarily designed to perform exception reporting. The data required for interrogation can be extracted using a series of criteria in an equation. The selected data can be written to the users PC in most of the common formats available. This package is fully Windows orientated providing a familiar feel to Windows users. The data extracted by Prospector will need further interrogation by a package such as Idea. This tool would probably be used by the CAATTs specialist to provide an auditor with the data they require, in a format compatible with their chosen interrogation software.

Sage Sterling

Sage is a file interrogation package designed for the smaller accounts application (usually PC based).
This software is able to import and export all of the common PC based file formats (Lotus, Dbase....). The auditor can perform exception reports, totaling, sampling but not stratification.

Sage will perform most of the usual accounting functions such as Bank statements, Accounting History, VAT return analysis. The auditor is able to insert journal entries and apply them to the audited body's accounts.

CA Panaudit Plus

This file interrogation software can operate on either the PC or the Mainframe. If the auditor chooses to work on the mainframe the software will create JCL code for the mainframe and send any output back to the PC. The software performs a wide range of auditing functions including:

- totaling;
- sampling;
- stratification;
- file comparison;
- ageing;
- The software also has a feature to translate plain English statements into equations.

Go to the CA Web site for support information.

Web Resources and Selected Articles on CAATTs

AuditSoftware.net
AuditNet Computer Assisted Audit Tools and Techniques (CAATT)

CAATTs Ideal for Efficient Audits by: Mark D. Mayberry, CPA.CITP, CISA

Computer Aided Audit Tools From Wikipedia, the free encyclopedia

Fraud data interrogation tools: Comparing Best Software for Fraud Examinations by Rich Lanza, CFE, CPA, PMP

Google Search on CAATTs