CSI
INTERNATIONAL

White Paper
Data Mining for the IT Department

The chances are that your organization has a tool that could be used for data mining, such as a report-writer or 4GL. Although used extensively in the business side of most operations, these tools are often neglected in the IT department, and yet they can be an invaluable aid in many areas.

It is an established fact that there are never enough hours in the day in IT. There is production work to run, development and maintenance to carry out, problems to solve, trends to spot, and information to analyze. A data miner can increase the efficiency of these tasks by filtering excess data from files and reports, building subsets of files for testing and fault diagnosis, and so on. In this article, we will examine a few of the ways that a data miner can help your environment function better.

Common Characteristics of Data-Mining Jobs

As mentioned in the white paper "Data Mining—Rapid Access to Buried Information," data-mining tasks share a subset of a number of common aspects. These include urgency, short shelf life of the results, incomplete results being better than none, ad hoc scheduling, a need to join unconnected data sources, and the security of the output. Not every data-mining task has all of these characteristics—but every job has some of them.

The IT Department and Data

The IT department is in the unenviable position of being responsible for all of an organization's data but owning a very small percentage of it. Data is one of a corporation's most valuable assets and usually belongs to the user departments that maintain it. The IT department owns a mountain of data—RMF, SMF, utilization and incident reports, and so on—that frequently consists of records of no great interest (outside IT).

IT is given the responsibility of keeping all data secure while developing applications to work with that data as quickly as possible. IT is also responsible for the secure transmission of data to remote locations.

Data Mining Within IT

Modifications to applications require testing, yet test data files rarely reflect the current state of the production application, not to mention the conditions being added by the most recent change. This disconnect is easy to explain because in most cases it is hard work to make sure that every condition is represented within test data; therefore, test files are allowed to get out of date, or we spend excessive time manually creating test data.

A data miner can make the job of creating test data that reflects reality much easier. Simply by copying a number of records from a production file, you can get a test file that does reflect reality. This method may introduce security concerns depending on the data being copied. A data miner can help address security concerns by extracting records from a production file and incorporating random data into sensitive fields such as addresses, phone numbers, names, social security numbers, etc., without modifying the remaining fields. To make the file even more useless to unauthorized viewers, the numeric data can also be manipulated by the data miner to change its values. An example of this latter function is to exclude records with values outside a certain range. (Bill Gates's bank account would be pretty easy to identify by its balance.) The data miner could double the balances for all records with account numbers ending with an odd number and halve the balances for the remaining records. The possibilities are limited only by the imagination of the test data creator!

Security is still a consideration because only a limited group of people generally have access to production data. They can write standard "data-fudging" routines and incorporate them in every data-extraction job that they do for programmers. The data-fudging routine could be a very simple set of commands such as the following:

Example 1

```
IF FIRST-NAME > "THOMAS"
      LAST-NAME = "LINCOLN"
ENDIF
```

or

```
ACCOUNT-BALANCE = ACCOUNT-BALANCE + (AGE * CHILDREN)
```

In these examples, we are using completely unrelated fields to change the value of potentially sensitive fields. A few or dozens of such commands can be strung together to change the value of every sensitive field if required. If the fudging is kept in a source library member called `FUDGEIT`, the entire copying job would look like the following:

Example 2

```
COPY                                       [Copy the input file to the output file]
INPUT=VSAM  FILENAME=PRODFIL
OUTPUT=VSAM FILENAME=TESTFIL
INCLUDE     LAYOUT                        [Member with record layouts for both files]
INCLUDE     FUDGEIT                       [Member with commands to fudge the data]
```

Programmers typically neither want nor need an entire production file to do unit testing; however, they may also want to make sure there are records in the file that meet the specific conditions to be tested. So Example 2 could also contain a maximum record value, commands to make sure some records have the required conditions, and `ONLY` or `SKIP` commands to select or reject records that meet certain conditions. For example, we could add these commands:

Example 3

```
IF ZIP > '40000'
      BALANCE = 0                 [Set our test condition in some records]
ENDIF
LAST = 500                        [Stop after 500 records]
ONLY        RECTYPE = "D"         [The only type of record we want]
SKIP        BALANCE < 0           [Don't include records with negative balance]
```

Using a Data Miner for Problem Determination

Much of the time spent in problem determination and correction is spent on finding the record that has caused the problem, not correcting the problem with the code. A data miner can help reduce this time significantly by enabling a programmer to create a smaller extract file containing only the record(s) they want or perhaps a few records on each side of the offending one. In this example, our application program has ABENDED because a numeric `BALANCE` field in a record has the word `HUGGINS` in it, causing a data exception. Rather than dumping the whole file or rummaging through control blocks to find the offending record, a simple data miner job is executed. The output has been trimmed to fit within the margins:

Example 4

```
       CONNECTIVITY SYSTEMS INC.
DATAMINER INPUT SCRIPT REPORT - VERSION 7.1
===============================================

      1     DUMP
      2     INPUT=VSAM FILENAME=VSMKSDS
      3     BALANCE  47   7   C
      4     ONLY BALANCE = 'HUGGINS' which produces this report:-
RECORD NUMBER - IN=      72,138, OUT=          1 LRECL=    223
    0  F1F8F6F8 <snip> D46ED1D6 C8D54040  *18682629RECNO>00000065...eJOHN   *
   20  40404040 <snip> C8E4 C7C7C9D5 E240 *          ...cHUGGINS           *
   40  40404040 40404040 40404040 etc etc
```

So in a matter of seconds, the data miner located the 72,138th record in the file as the problem record. This is, of course, a very simple example—we could have put together a number of commands to find the record we wanted, to perform a calculation on the key of the offending record, to reposition the pointer several records before the offending one, to dump the next 20 records, etc.

Pre-Vetting Test Data

How often have you heard (or used) the excuse that the test data file didn't contain the condition you wanted? A data miner lets you verify that the conditions you need do actually exist. This checking can be done from a single file or by joining several files together, using a key from the first file as a link to a second file. For example, if your latest program modification is meant to prevent orders from being placed by a customer whose account is closed, you can verify that such a pair of records actually does exist in your files. Then your test job will be sure to test the conditions that it is meant to. You can even get a listing of the contents of those records, as a simple hex dump of the records or a formal report.

Status and Performance Reports for IT

How many reports get run in your IT shop for the IT department itself? There are probably dozens, and it's not unusual, to quote Tom Jones, for them to contain far more information than you need. There is an old saying that "the value of a report is inversely proportional to its size."

If I may digress, my first job out of college was as a programmer with a stock broker. The application on which I worked produced a 100-page report every day on seven-part paper, and I was intrigued to see what happened to it. I delivered it to the user department one day and asked them to show me what they normally do with the listing. A lady took the report, ran her finger down the right-hand column, and then cut the 30 lines with the largest values out, taped them to a large sheet of paper and ran off five copies on the photocopier. They simply wanted to know the 30 funds that were giving the best returns. It was a matter of half a day's work to add a sort to the job and print it five times.

We still see this over-production of reports in the IT department every day in the form of network outage reports full of columns of zeroes because we have had no network outages, which leads to the occasional nasty figure being hidden.

A data miner could easily run these reports, omitting the lines with no useful information to report. A simple `ONLY` or `SKIP` command would be used to ensure that only pertinent information was reported. A data miner may also be used to whittle down reports produced by third-party software products such as performance monitors. All that is needed is to either pass the product's input file through the data miner to remove all the zero records or to spool the report to disk and then read it back into the data miner, printing only those lines where the statistics you are interested in contain meaningful data.

Conclusion

Although data-mining tools are often purchased by user departments to analyze business data, it is equally often the case that the IT department can make as much, or even more, use of them than the end users. So there really can be such a thing as a free lunch after all.

A Word About the Examples In This Article

All of the examples in this article have been created with Data-Miner from CSI International. For the sake of brevity, field definitions have been left out of the examples; they can be supplied either in shorthand form (for example, `CUSTNO 1 8 C`, meaning an eight-character field starting at byte `1`) or included in the job from a COBOL copybook. Explanatory comments are in square brackets.

For more information, call 800.795.4914 or visit the Website: www.CSI-International.com