Need
Example Data?
Sample The Web

FEATURES | JANUARY/FEBRUARY 2009

DEPARTMENTS

From the Inside

BY NATHAN RECTOR

Innovate or your application will die a slow death. Harsh words, but you'll start seeing more and more CIOs, CTOs, and even CEOs demanding more and more from their existing systems and business applications.

Most companies rely on their business software and systems more than they rely on their employees. As companies start cutting back on costs, they will look at the IT department, and see 80% of their IT budget applied to maintenance of existing systems, with the remaining 20% on new innovations or R&D.

This is the 80/20 rule or paradigm. While this is valid, it is also a trap that many IT departments fall into. When times get tough, they start focusing more of their budgets into the maintenance role and forsake innovation. While this works well for the short term, many times it becomes the long term plan as well, causing company business systems to fall farther and farther behind the requirements of the business.

With the way things are in the economy, I expect to see many companies will start looking at the IT budget, but I don't expect them to do the wholesale cutting they did during the dot-com bust. During the dot-com bust, many companies started to cut IT budgets, but quickly found that they could not do it without adversely impacting the productivity of the rest of their business.

Instead, the companies that invested in their IT departments, and specifically in new software tools and innovation, found that they could run their companies with less people and more software. While this concept only goes so far, many existing business applications severely lack the innovations in key areas due to the focus on the 80/20 paradigm.

Some of this is due to lack of planning, bean counting, and even lack of knowledge that something can or cannot be done within their business systems.

Another reason is the "I can do it better and cheaper" concept that is so prevalent in the MultiValue database market.

Since the MultiValue database environment is so efficient and effective in developing applications that don't take so much time or resources, we have a tendency to overlook all the tools that exist in the marketplace that have already solved many of the application problems and user demands.

Or we look at the tools and say to ourselves, "I can do that better and cheaper". And we spend the 20% of our IT budget developing a tool that we could have purchased cheaper. Then, we spend a part of the 80% that should be spent on business systems maintaining tools developed in-house, when a third-party tool will always end up with more features and flexibility than we ever have time to program into the "in-house developed" tool

Take a look at the ROI of developing a tool vs. purchasing a tool — 60 to 200 hours of your time developing the tool is likely to cost you around $4,000 to $14,000. Depending on the tools, you have already spent the license costs, plus a few years of support fees.

Now, look at the maintenance costs if you write the tool yourself — 20 to 500 hours a year on maintenance and add new functionality as needed. Again, ends up being around $4,000 to $35,000 a year. Most yearly maintenance costs for these tools are less that this.

Using third-party tools is well worth it, and it allows you to spend more time on focusing on improving and innovating business systems instead of creating new tools.

Improving and innovating business systems is not solely a focus of fixing the bugs in the existing systems, but building the business software framework that can benefit from the new technologies that your CEO is demanding you put to use. More and more companies are looking at SaaS tools and applications to enhance the productivity of their business, as well.

SaaS business applications like SalesForce.com are becoming more and more prevalent in business applications for the same reason that third-party

Getting Real

BY ROSS MORRISSEY

Some of you will laugh at this one. Data? You've got too darn much of it. But if you're working independently, or you're a software developer, one of the challenges of learning, testing, and evaluating software — especially reporting tools — is the lack of good example data.

Fake data might help you test or demonstrate the user interface but it fails on a number of counts. Real data has a number of advantages, but has its own challenges. This article will look at the trade-offs, then show how to build a set of real example data that can be will be used to demonstrate reporting solutions.

There are two main kinds of fake data — generated data and obfuscated live data. Generated data is produced by a program according to rules that simulate the kind of data we expect to encounter. These programs can quickly become quite large as the data becomes complex. Generating a set of sales transactions for random products to random customers is close to the limit of this technique. By basing values on typical numbers seen in live data, a more realistic data set can be created, but it will not tell a story.

Obfuscated data takes a subset of live transactions and masks identifying features to avoid privacy and confidentiality issues. One major problem with this approach is in selecting a subset of live data to obfuscate. Selecting transactions during a short time frame might mask useful seasonality trends, while selecting certain classes of transaction or geographic areas can make it difficult to capture the richness of the source data. The obfuscation exercise can also take a tremendous amount of time if realistic sounding customer names or product names are required. Another problem is that this data is static. The dates associated with the data soon look stale unless they are constantly refreshed.

Another approach is to obtain some live data from a domain that most people are familiar with. The problem with sales data is that the product being sold will resonate with a very small audience, and they will probably be trying to figure out whose data it is, rather than focusing on the software.

For this article and subsequent articles, the Seattle Fire Department's Real-Time Dispatch data will be used. It has several advantages: everyone knows what a fire

> *The problem with sales data is that the product being sold will resonate with a very small audience, and they will probably be trying to figure out whose data it is, rather than focusing on the software.*

truck is and does; the data is available up-to-the-minute; it is public; and it's quite rich, with geographic, equipment, and incident dimensions. Although almost nobody will have a direct connection to the data, it will tell a compelling story that every audience can understand.

The City of Seattle hosts a web form (figure 1) that queries an Oracle database to provide a list (figure 2) of fire department dispatches in Seattle (intl-spectrum.com/s1008). The format of this page (figure 3) lends itself to reverse engineering to extract the incidents for each date back to late 2003. A UniVerse program to extract this data and create a file structure in UniVerse ran in a couple of hours on UniVerse PE.

The program is quite straightforward, with any special logic contained in about a dozen lines (figure 4). The URL is constructed from the address that the original form created, with an embedded external date and a hard-coded direction. The UniVerse CallHTTP capability is easy to use, but wget or Perl has been used for this sort of work in the past.

A quick way to parse through HTML documents is to convert the angle-bracket delimiters to UniVerse delimiters and use REMOVE to step through the string. This will alternately provide tag information and content of the

Fig. 1 Real-Time 911 Dispatch Form

Fig. 2 Real-Time 911 Dispatch Results

```
<tr id=row_2 onMouseOver='rowOn(row_2)' onMouseOut='rowOff(row_2)'>
    <td class="closed" width=16% valign="top">11/18/2008 12:03:17 AM</td>
    <td class="closed" width=10% valign="top">F080106228</td>
    <td class="closed" width=7% valign="top"></td>
    <td class="closed" width=17% valign="top">E40 M1</td>
    <td class="closed" width=36% valign="top">8242 20th Av Ne</td>
    <td class="closed" width=14% valign="top">Medic Response</td>
</tr>

<tr id=row_3 onMouseOver='rowOn(row_3)' onMouseOut='rowOff(row_3)'>
    <td class="closed" width=16% valign="top">11/18/2008 12:04:42 AM</td>
    <td class="closed" width=10% valign="top">F080106228</td>
    <td class="closed" width=7% valign="top"></td>
    <td class="closed" width=17% valign="top">M10</td>
    <td class="closed" width=36% valign="top">8242 20th Av Ne</td>
    <td class="closed" width=14% valign="top">Medic Response</td>
</tr>
```

Fig. 3 Real-Time 911 Dispatch Results – Source Excerpts

```
URL = \http://www2.seattle.gov/fire/realTime911/getRecsForDatePub.asp?\
URL := \incDate=\:CHANGE(THIS.EXTERNAL.DATE,"/","%2F"):\&rad1=asc\
Ret = createRequest(URL,'GET',HTTP.HANDLE)
RET.STATUS = submitRequest(HTTP.HANDLE,'','',HTTP.HEADERS,HTTP.SOURCE,HTTP.STATUS)
RAW.DATA = CONVERT("<>",@AM:@AM,HTTP.SOURCE)
THIS.ITEM = ""
LOOP
    REMOVE THIS.ELEMENT FROM RAW.DATA SETTING DELIM
    BEGIN CASE
        CASE (THIS.ELEMENT[1,10] EQ "tr id=row_")
THIS.DAY.OFFSET = FIELD(THIS.ELEMENT[11,9]," ",1)
        CASE (THIS.ELEMENT[25,2] EQ "10")
REMOVE THIS.INCIDENT FROM RAW.DATA SETTING DELIM
THIS.ITEM<DISPATCH$INCIDENT> = THIS.INCIDENT
        CASE (THIS.ELEMENT[25,1] EQ "7")
REMOVE THIS.LEVEL FROM RAW.DATA SETTING DELIM
THIS.ITEM<DISPATCH$LEVEL> = THIS.LEVEL
```

Fig. 4 Sample code from LOAD.FIRE

BY CHARLES BAROUCH

Business Tech:
Becoming a
Software Vendor

Part 1

Is There Still A Market For Commercial Software?

The day the first TV broadcast aired, all the movie theaters closed. The first photographic flash signaled the death of painting. Likewise, the first time free software showed up, every commercial developer closed up shop.

Obviously, none of that happened. Free software and its close cousin, included software, aren't new. They have been with us since the first computer was sold. Manufacturers cared about hardware, so they saw software — even operating systems — as an add-on. Software was just something needed to move product; hardware was the real product.

Now TV, photography, and free software all have had an impact on their respective landscapes. When TV gained popularity, it changed the rules, roles, and importance of movies. It did not destroy them, but it did force them into a less primary position. From that perspective we can see that commercial software has to be more blockbuster and less formulaic — which means that the first challenge for the software developer is to find a category worthy of people opening a wallet. Actually, that's only one of two routes to our goal.

The other route, is best summed up by my friend, Patrick Clancy, who says, "Free software is like a free kitten. You still have all the costs associated with adopting a technology." While he means it as a caution to adopters, we can see it as a revenue path: free software with paid training, paid support, paid add-ons, etc.

F[r]ee Software

When was the last time you, yourself, paid for software? Same question, only this time say support or training? Before you can sell something to others, you need to understand the willingness to pay for software. What makes a product worth buying? When is free more expensive or less productive than paid?

As for myself, I use both kinds of software. This article is being written using a commercial product. It will be mailed from a freeware mail client. The magazine you will read it in is a free subscription. The ads in the magazine offer all sorts of paid software and services. F[r]ee software, the mix of paid and unpaid, is the way of the world. Blending helps us succeed.

Use the Balance Sheet, Luke

So, which way should you go with your killer idea? This is where we remember the first word in the title of this series: Business. The frightening thing for many of us is that the "Tech" part is not the most important part of developing and selling software. We will cover that aspect, but we have to talk business first.

Assuming your software falls into an existing category, you should look at what your competition offers and how your product will be different. You'll notice that I said *different*, not better. When I buy software, or a car, or anything, I rarely look for the objective best. I look for the best fit for my specific needs.

If one product is better but the features which differentiate it have no value to me, then the objectively lesser product might win me over by having a lower price, more aggressive upgrades, a better company behind it, or perhaps by having a lesser feature which happens to save me a lot of time. For example, Photoshop is much better than the photo software I used for years. However, the one I used had a drastically lower learning curve and a really simple automation tool. My choice of software wasn't even in the top ten objectively, but my work got done. The software was clearly worth the money.

Do I still use that software? No. Did I switch to Photoshop? No. I switched to a different also-ran. Am I advocating the creation of software that isn't the best of the best? Yes. Be-

cause software I can justify the cost of, software which has the features I need, software I can get up to speed on quickly — that's what I'll spend my money to buy. That's the point where I open my wallet.

That last bit may hurt your tech pride, but it makes business sense. The fact is, I do buy the more expensive software when it will save me enough time. I never buy the more expensive software just because it is cooler or has more features.

Think about this example. If a consultant costs $150 an hour and a $300 software package will cut the project by three hours, it is already worth the cost. The benefits are similar with staff projects. A free package which will increase the project by the five minutes it takes to learn it is a net loss compared to spending $300 on the right solution for you.

Break ~~Neck~~ Even Speeds

All business projects should start with a break-even chart. This is a magical device which allows you to estimate how many you need to sell to make back what development, marketing, et al costs you. A properly constructed break-even chart will tell you that no project is ever worth starting — there are exceptions, but not many. The ratio of success and failure on software projects leads you to similar conclusions. So, I guess we can stop here. Except...

Statistics on open source projects tell an interesting story, a story which explains successful commercial software as well. The brief version is: *software developed for a real client who has a real need tends to survive and grow.* It seems that the best projects are never developed as products. They are developed as specific solutions and then generalized into products.

Here's the inherent challenge: if successful software starts as a project instead of a product, how do you make sure you can legally sell the product if you did the project for someone else? Remember, if you start out by violating copyright, you can find yourself unable to protect yourself from theft. So, besides being wildly unethical, stealing rights is impractical. Be sure of your rights.

Back on the break-even chart, if the product started as a paid project, then the version you have in your hands is effectively "free" in that it has already

---

If one product is better but the features which differentiate it have no value to me, then the objectively lesser product might win me over by having a lower price, more aggressive upgrades, a better company behind it, or perhaps by having a lesser feature which happens to save me a lot of time.

---

been paid for by the client, assuming you still retain the rights and sources. That means that even as the developer, you are dealing with free software — your own. If the software is free, what goes on the break even-chart? Doesn't that mean we have 100% profit margin?

Mark of Zero

"He who has a thing to sell

And goes and whispers in a well,

Is not so apt to get the dollars

As he who climbs a tree an hollers."

-Unknown

Climbing trees costs money. You might think of companies like Yahoo and Google as technology companies, but that isn't where their money comes from. Google makes a fortune from ad revenue. Before you can place an ad, you have to develop an ad, or better yet, an ad campaign. Of course, you first need a company, a logo, a web site, a phone number, an address, and so on. There are many things to put on that break-even chart.

Funny thing about that phone number: people expect someone to answer it, or at least return calls. Same with your corporate e-mail, questions on your message boards, posts to your LinkedIn/Yahoo/Google groups, and every other communication venue. Staff likes to get paid. Staff likes to have enough equipment to do their jobs. They may need a place to sit, a phone, even a computer or two.

One of the ways to control those costs is to limit the initial methods of contact. Another solution is to be upfront with customers and prospects on expectations. For example, when I go to the store, they post hours on the door. This has a double value: (A) they promise to be there 9:00 AM to 5:00 PM weekdays, and (B) to promise to not expect them there any other time. In technology, we call this an SLA (Service Level Agreement). If you tell clients that the phone number is voicemail only, and you agree to a four hour response window during business hours, you can control how much staff you need and how much of the week needs coverage.

When you build out all of the costs, you may find that the easiest part of a software business is the software. This leads to the idea of spreading the risk and not just the work. Partners, unlike staff, don't expect to be paid right away. The same goes for investors. Please remember that people who share risk expect to have a voice in the product, the business, and all the other aspects of your plan.

Creating a Logging Client

BY BRIAN LEACH

One of the difficulties involved in any client/server or web project, and one that has recently raised its head again on the U2 Users list, is the problem of working out just what exactly is going on at any moment.

Best practice recommends that you maintain all your logic in Basic subroutines so that they can be easily tested and documented in advance and in isolation, preferably before commencing on writing the client. In that way, you know that you have a strong foundation on which to build.

Inevitably, however, there may be circumstances in which those routine may fail to produce the expected results—something you thought was initialized is not, or the user has found a path through the program that you did not anticipate. At this point, isolated testing on the server may not be enough to fully replicate the problem.

The problem is that, for most of the forms of middleware available, the database server is treated as a black box. A call to a subroutine over, for example, the IBM UniObjects API does not permit any form of interaction to take place during the call. If the subroutine requests input half way through or raises an error message, there is no way to access these.

The usual approach is to set some kind of logging flag for testing to have the server log all its activity to a file. The difficulty raised is the analysis afterwards, especially if you have to look through several log files together to see what the client and server are both doing, and to try to coordinate these. Also, the critical events may be buried deep inside a huge log that makes it difficult to identify.

To further complicate matters, the tools I oversaw in my previous company used a variety of different client languages, often in layered solutions, and supported a range of different databases. So to help the support process, we created a global logging client.

The aim of the logger client was to allow us to watch logging messages raised from all the various languages in real time, so that we could see the interaction between these components and the server. The display could be paused or sent to a single log file. Since I now have my own client/server tools to support, I thought the time was ripe to write a new global logger for myself.

Client Side Logging

For the client components having the logger running on the client makes it relatively easy to pass logging messages across. The obvious choice was to build the logger as a COM server. COM is an older, but useful technology that enables Windows clients to remotely instantiate classes exposed by a program or library. Visual Basic, Delphi, .NET, and others support COM, so it is useful for cross-language communications. (Fig. 1)

Clients can first check to see if the server is running by using the Windows FindWindow() function to try to locate and return a handle to the logger's main window. Only if the logger is running, will the clients try to get an instance of an exposed Logger class and to call a Log(Text) method to update the main display in real time and optionally build a consolidated log file. The syntax for calling the Log() method will vary by language. Figure 2 shows some Delphi snippets for calling the Logger.

I decided to use Delphi rather than C# for the logger itself because it compiles into a single, stand-alone executable that can be easily copied to a machine and, further,

it allows me to use the excellent open source Indy socket components to support the server side logging.

Server Side Logging

For the servers to send logging messages the clear choice was to use sockets. UniVerse, UniData, OpenQM, and D3 all offer socket support, each providing a set of functions that encapsulate and simplify network programming.

For UniVerse and UniData, the Basic language has been extended to offer a function-based socket interface. There is one important restriction on using this—if you open a socket from a phantom, it will consume a license. Apart from that one annoyance, the socket API makes a good job of encapsulating regular TCP/IP communications in a platform-neutral and syntactically familiar fashion.

There are two forms of sockets that can be manipulated in the basic socket API: a server socket and a client socket. A server socket listens for and accepts incoming connection requests. A client socket initiates a conversation, and is the more generally useful of the two.

Server sockets are usually associated with service provision (e.g., an FTP server) that handle multiple concurrent requests: neither UniVerse nor UniData permit this. Once established, both socket types allow for bidirectional communication and can be used in the same way.

The client socket connection is performed by the openSocket function (fig. 3). The *host* can be either an address or host name, the *mode* specifies either blocking or non-blocking and the handle returns an identifier to the underlying socket. If the connection succeeds, the *returnValue* is set to zero following C conventions.

A socket connection is like a phone conversation: once the socket is open, both participants are free to talk and listen without concern about the layers
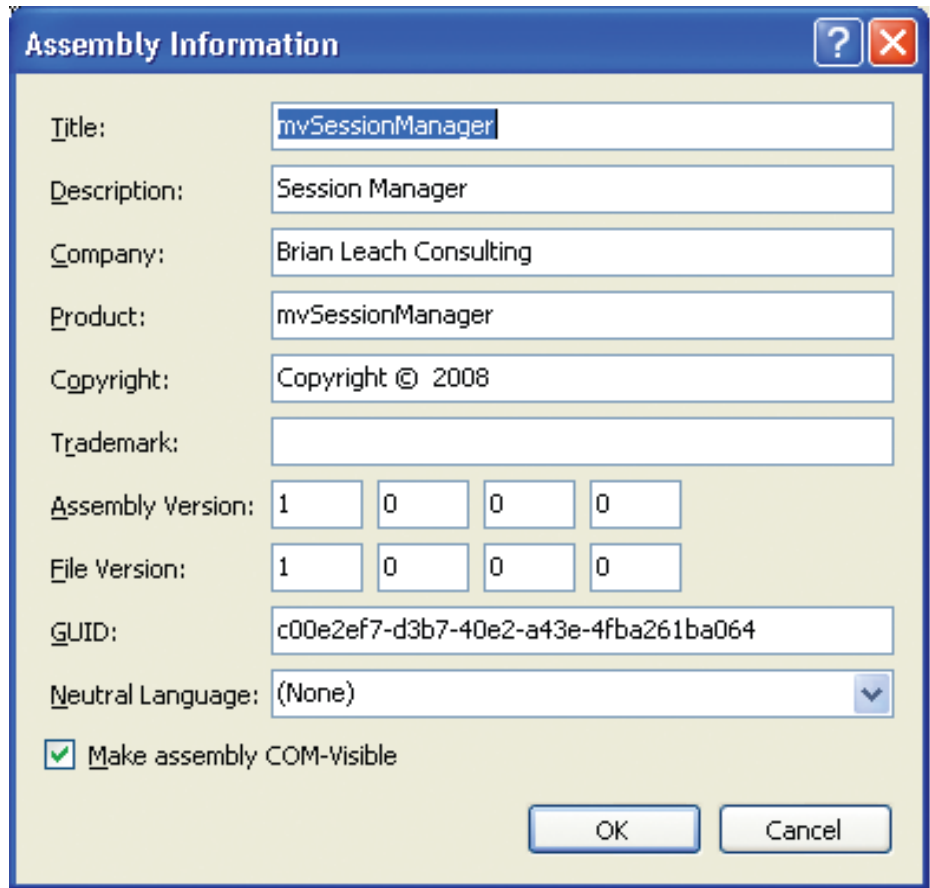
Fig. 1 COM enabling a .Net application

```
Function StartLogger:Boolean
Begin
   If IsLoggerRunning Then Begin
      try
         Logger := CreateOleObject(LOGGER_NAME);
         Result := True;
      except
         Result := False;
      end;
   End;
End;

Function Log(Text: String)
   Logger.Log(Text);
End;

Function IsLoggerRunning: Boolean
Var
   Handle: HWND;
Begin
   Result := true;
   try
      Handle := findWindow(PChar(FORMNAME),PChar(FORM_TITLE));
   except
      Handle := 0;
   end;
   Result = (Handle <> 0);
End;
```

Fig. 2 Calling the Logger from Delphi

```
returnValue = openSocket(host, port, mode, timeout, handle)
```
Fig. 3

CREATING A LOGGING CLIENT

of technology in-between (so long as it all works!). The logging is, of course, principally concerned with sending messages, which it does through the writeSocket function (fig. 4).

The *handle* refers to the socket handle returned from openSocket, and *data* in this case holds the text sent to the logger. In blocking mode, the function will attempt to send all of the data before returning. In non-blocking mode, it will send as much as possible, setting *bytesWritten* to the size of data actually sent.

Although not required for the simplest of loggers, there is a mirror function used to receive data (fig. 5).

The readSocket function will attempt to read up to *maxSize* number of bytes from the socket and place the results into data. The actual number of bytes read is returned in *bytesRead*.

At the very end once the socket is no longer required, the socket should be explicitly closed by calling the closeSocket function (fig. 6).

```
returnValue = writeSocket(handle, data, timeout, mode, bytesWritten)
```
Fig. 4

```
returnValue = readSocket(handle, data, maxSize, timeout, mode, bytesRead)
```
Fig. 5

```
returnValue = closeSocket(handle)
```
Fig. 6

Figure 7 illustrates a simple server side logging subroutine. A named common block is kept to hold the logging status and the socket handle. The result is a simple logging mechanism that makes debugging client/server operations much less frustrating (fig. 8). **IS**

BRIAN LEACH is an independent IBM U2 and MultiValue consultant based in the UK and the author of a series of books on IBM UniVerse. You can find out more at www.brianleach.co.uk.

```
SUBROUTINE stdLog( Text )
COMMON /LOGGER/ LOGINIT, LOGGING, LOGSOCKET, LOGCLIENT

EQU SOCKET.OK To 0
EQU MODE.BLOCKING To 1
EQU MODE.NONBLOCKING To 2
EQU CRLF To CHAR(13):CHAR(10)

If UnAssigned(LOGINIT) Then
  LOGGING = @False
  LOGINIT = @False
End

* Read the location of the logger client
If LOGINIT = @False Then
  LOGINIT = @True
  Call stdReadIni("blCONTROL", "LOGGER.INI", "Logger", "LogHost", LogHost)
  Call stdReadIni("blCONTROL", "LOGGER.INI", "Logger", "LogPort", LogPort)
  Call stdReadIni("blCONTROL", "LOGGER.INI", "Logger", "LogClient", LOGCLIENT)

* now try to connect - if the logger is running
  Ok = OpenSocket(LogHost, LogPort, MODE.BLOCKING, 0, LOGSOCKET)
  If Ok = SOCKET.OK Then
    LOGGING = @True
  End Else
    LOGGING = @False
  End
End

* We are logging
If LOGGING Then
  Packet = "[":LOGCLIENT:" ":@UserNo:"] " : Text : CRLF
  BytesToSend = Len(Packet)
  Ok = writeSocket(LOGSOCKET, Packet, 0, 1, SentBytes)
  If Ok <> SOCKET.OK Then
      * error on socket, turn logging off (i.e. logger closed)
    LOGGING = @False
  End
End
RETURN
```

Fig. 7 Simple logging subroutine

Fig. 8 Server side logging for mvScript pages

GETTING REAL

tags. A CASE statement can be used to identify patterns in the generated HTML that correspond to particular data values. The HTML source example shows a width attribute that contains unique values for each data type, which is very helpful.

Complete source code and supporting dictionaries and one month of data are available via intl-spectrum.com/s1009.

The DISPATCH file with data from 2004 through 2008 contains about 450,000 items which is a good size to test performance of reporting tools (figure 5). There are a number of dimensions that can be constructed from the incident, unit, and address information.

The incident descriptions fall into a number of categories that can be created manually (figure 6). Address information can be parsed to find the street name and the directional por-

tion of the address that defines areas in Seattle (figure 7). Each dispatch can contain multiple units — for example E40 and M1. Two simple I-descriptors provide Unit type and trip count: OCONVS(UNIT,"MCA") and NES(UNIT,REUSE("")). Additional dimensions could easily be created for hour of day, season, to support different types of analysis.

In the next article we'll look at the data in depth and what kinds of stories we can tell using traditional MultiValue reporting. **IS**

ROSS MORRISSEY helps MultiValue shops with Business Intelligence challenges. If you're stuck, can't get started, or want to do more, contact ross@tantiva.com.

```
SORT DISPATCH WITH DATE EQ "11/18/2008" 02:39:26am  19Dec 2008  PAGE    1
DISPATCH.. DATE.... TIME. INCIDENT.. UNIT. ADDRESS........... TYPE............

14933*002  11/18/08 12:03 F080106228 E40   8242 20TH AV NE    MEDIC RESPONSE
                                     M1
14933*003  11/18/08 12:04 F080106228 M10   8242 20TH AV NE    MEDIC RESPONSE
14933*009  11/18/08 12:34 F080106231 M1    517 3RD AV         MEDIC RESPONSE
                                     A5
14933*012  11/18/08 12:39 F080106233 A2    1811 EASTLAKE AV   MEDIC RESPONSE
                                     M10
14933*013  11/18/08 12:40 F080106234 E33   4444 S THISTLE PL  AID RESPONSE
```

Fig. 5 UniVerse DISPATCH file sample

```
SORT DISPATCH.TYPE BY CLASS
DISPATCH.TYPE TYPE......................... CLASS...

1             MEDIC RESPONSE                AID
105           AID RESPONSE FREEWAY          AID
110           MEDIC RESPONSE FREEWAY        AID
17            MOTOR VEHICLE ACCIDENT        CRASH
2             MOTOR VEHICLE ACCIDENT FREEWAY CRASH
70            MULTIPLE CASUALTY INCIDENT    CRASH
```

Fig. 6 Sample Incident Types

```
LIST DISPATCH.ADDRESS ADDRESS STREET AREA CITY
DISPATCH.ADDRESS ADDRESS.................. STREET............ AREA CITY...

17416            NE 104TH ST / 17TH AV NE  NE104TH ST         NE   Seattle
67047            6262 NE RADFORD DR        NE RADFORD DR      NE   Seattle
43032            4063 34TH AV S            34TH AV S          S    Seattle
65446            3848 37TH AV SW           37TH AV SW         SW   Seattle
38229            1128 25TH AV              25TH AV            C    Seattle
```

Fig. 7 Address Structure

ERP Systems and the MultiValue Advantage

BY CANDI HART

Today when people say **ERP** they think **SAP, PeopleSoft, JD Edwards, Oracle,** and many other big names. But that's all marketing.

My initial entry into the MultiValue world was in 1979 when I started working for a company that had an early version of ERP software. It wasn't sold as *Enterprise Resource Planning* software; the acronym ERP hadn't been coined at that time. It wasn't until around 1990 that the term was introduced by the research and analysis firm, Gartner, Inc. The term came about as the evolution of Material Requirements Planning (MRP) to Manufacturing Resource Planning (MRP3) and finally the generic term Enterprise Resource Planning. SAP is generally given the credit for inventing the market and some reports say that they still have the biggest market share, but I'm not sure if that survey included the numerous MultiValue applications that have continued to be successful since the 1970s.

Let's talk about what ERP is, the benefits, the problems, and what advantages our MultiValue packages have over the big guys.

Just What is An ERP System?

According to Wikipedia, ERP is an enterprise-wide information system designed to coordinate all the resources, information, and activities needed to complete business processes. The core of ERP is a common database with several modular applications that interact. In a typical manufacturing business, there are modules which handle the following functions:

- Finance/Accounting – includes General Ledger, Accounts Receivable, Accounts Payable, and Asset Management

- Order Entry

- Inventory Control

- Manufacturing – includes Bills of Materials, Routing, and MRP

- Purchasing

- Warehouse Management – includes Shipping and Receiving

- Sales Reporting

At a minimum, to be considered an ERP solution, a system must provide functionality and interaction between two or more modular applications, all of which have inputs to and draw outputs from the common database.

Though MRP originated out of a manufacturing environment, today ERP provides best business practices in non-manufacturing businesses like retail, transportation, and healthcare. And it is used in for-profit, non-profit, and government organizations. The concept of ERP is the integration of all basic business functions of an enterprise. CRM (customer relations management) and HR (human resources) have been added as standard modules, and additional functions are continually being integrated in this way.

Let's study an example. Since the concept originated out of MRP, let's look at a simple toy manufacturing enterprise and see how the ERP software connects the modules.

The Inventory Control module: This application enables input of part numbers and descriptions for the models of toy trucks in the company's product line. The part numbers assigned here are used throughout all modules.

The Order Entry module: This application enables input of an order. When the company receives an order for 10 toy

trucks, the part numbers assigned in the inventory control model are used.

The Accounts Receivable module: This module contains the customer's credit information. The order must be approved through access to this information before it can continue to manufacturing.

The Manufacturing module: This application establishes a Bill of Materials (BOM) that indicates what parts are required to make one toy truck. For our example, we will state that a toy truck requires (1) wooden chassis and (4) wheels.

A "router" is also set up in this module that defines the steps that must be done to build this part. It takes one day for assembly and one day to package the toy. When MRP is run it recognizes the order created through the order entry module and calculates a need for ten chassis's and 40 tires. Then, using the lead times established by Inventory Control, it calculates when the chassis and tires must be ordered.

After taking into consideration the routing steps, it creates a schedule for this order and indicates that the product can be available in three weeks. The scheduled completion date may be passed back to the order entry module so that customer service can notify the customer.

The Purchasing module: This software contains all the functions needed to enter purchase orders for the enterprise. For this order, they receive a *buy order* from the MRP calculation for the chassis and the tires, indicating that the chassis should be ordered today and the tires in 10 days.

The Human Resource module: This application contains employee information and regulations. Since the HR department is also responsible for hiring temporary employees, they may receive a request to have temp employees available to assemble the product

in 14 days based on the schedule created by MRP.

The Warehouse Management module: This module is responsible for all product coming into and going out of the warehouse. When the chassis and tires come in, this department must reference the purchase order created in the purchasing module and know that these parts are required in the manufacturing department. When manufacturing returns the completed toys back to the warehouse, they must have ac-

---

> *MultiValue ERP software vendors have been implementing and providing successful packages for more than two decades. They are experienced and fully understand the interface between departments. The software is easy and quick to modify.*

---

cess to the order and a shipping document in order to ship the product to the customer.

The Accounts Receivable module: This module must receive verification of shipment from the warehouse in order to generate an invoice to the customer. When the customer pays his invoice, the general ledger postings are made and the cycle is complete.

In our example, it is clear that it constitutes an ERP solution.

- It has several modules – typically applications for individual departments.

- It shares a common database.

- The software replicates business processes within the enterprises.

- There is communication across functions/departments.

- It attempts to *coordinate* the resources, information, and activities of the business enterprise.

Benefits of ERP Software

One Software Provider

The alternative to ERP software is a company that has several software applications—and sometimes separate computers. Since all modules in the ERP software are written by the same supplier, the user interface is the same over all modules. This means that an employee can transfer between departments, and the software's look and feel is consistent. The training on how to use the system would apply in any module.

The software support is simplified and reduces the number of software specialists the company must employ. Or, if support is external, it requires only one support contract and maintenance fee.

One Database

A single database means that the data is entered once at the point of origin, so the person entering the data is most familiar with it. Since that person/department is responsible for that data, it has a high percentage of accuracy. And since that data is accessed throughout the system, all reports are more reliable.

Because the data is entered into one database, the timeliness of the data is heightened. It doesn't require batch processes to update other systems before the information is available to other departments.

Modular Design

The modular design makes it possible for a company to purchase only the modules that it requires. Almost all companies require some finance application, but a retail company would not require the manufacturing module. The software makes it possible to run each module separately, but as mod-

ERP SYSTEMS AND THE MULTI-VALUE ADVANTAGE

ules are added they take advantage of the existing pieces of data already available. Implementation of each additional module therefore becomes easier.

Standard Business Practices

An ERP package is written to apply Best Practices to their processes. This is the way the software was designed to work and considered to be the most efficient method of performing those standard business functions. They are based on the most common and accepted way of doing business. In a company that has no standard procedures, in the process of implementation and training the personnel, consistent practices can be established. The software can prevent steps and approvals from being bypassed and ensure some system security. For example, the software can allow only specific personnel to approve a large order or override a price.

It will not allow a shipper to print unless credit has been approved.

Automated Communication

The more modules that are implemented, the more processes may be automated. This saves man hours and improves accuracy. In our manufacturing example, the credit department may have a desktop workbench which lets them know when a new order is entered requiring credit approval. When the order is released off the credit workbench, the shipper may print automatically in the warehouse—after checking for stock on hand.

Increased Efficiency

I remember when orders were printed as they were entered and hand carried to the credit manager for him/her to initial. Then they were delivered and held in the warehouse until the stock became available. The warehouse personnel would have to go check to see if the product was on hand because the order entry software didn't have access

to the inventory. The implementation of ERP reduced the number of lost shippers and increased the number of orders that could be entered and processed per day.

The Disadvantages of ERP Software

The purchase of ERP software is a large commitment both of time and money. There is little argument that it is of value—if the implementation is successful. The biggest problem that ERP packages have encountered is the failure of the implementation. That failure is most commonly not the failure of the software. It is because of underestimating the time required to retrain existing personnel and the resistance to changing existing workflow.

Independent and autonomous departments sacrifice control. They used to be able to research and choose any software package that best matched the way their department worked. Of-

FROM THE INSIDE

tools help your business. It costs less

to have someone else maintain it and keep it current. Then the business can have their IT staff focus more on keeping your business running instead of the interfaces running.

Many companies don't know, or are not told about, all the tools that exist to solve their problems. Now would be a good time to do a little research to see what tools exists that would make your job easier and cause your IT budget to be lower.

Remember, your CEO or company management is looking for ways to cut costs across the whole business, not just in the IT department. If the IT department can supply them with a tool that will save the company money, they will have no problem purchasing it. **IS**

-NATHAN RECTOR
President, International Spectrum
nathan@intl-spectrum.com

NEWSMAKERS

Fresno Pacific University Saves Time and Money Using Informer

Entrinsik talks with Jeff Jones as he discusses how Fresno Pacific University realized a significant return on their investment with Informer.

Tell me a little about Fresno Pacific University, Jeff?

Fresno Pacific University is the California Central Valley's only accredited Christian university, offering leading academic programs, ethical and spiritual development through traditional bachelor's degree, adult degree completion, master's, certificate and credentialing programs.

What is your role at Fresno Pacific University?

I'm the Institutional Research Coordinator at Fresno Pacific. My office does all the ad hoc, internal, and external reporting for our University primarily from our student information system. Consistent, accurate, and immediate data are imperative to our operation.

Describe a recent challenge you faced within your department.

We needed an affordable way to access this data that was secure, easy to use, and accessible for virtually anyone to create and customize reports in minutes without help from our database experts. The requirements we had were fairly straightforward — an affordable solution that works with our existing database to provide more sophisticated reporting than what already existed.

In what way did the capabilities of Informer address this challenge?

After searching among the available reporting options, we selected and implemented Informer. We've been using it for a couple years and now average about 15,000 reports — both scheduled and ad-hoc — on a monthly basis. Our users can now instantly create and customize reports using an intuitive browser-based interface.

Informer is able to perform real-time extraction of our live data significantly faster than we were able to in the alternative solutions because it does not have to move individual records to a data warehouse or cube. Best of all, the data is always accurate and up-to-date because there is no batch process to wait for.

And what results did you see from this approach?

Given the success we've had with Informer, we then decided to identify our return on investment in Informer. We quantified in minutes the time spent on reporting before implementing Informer and compared it with the time spent with Informer in place. After crunching the numbers, we realized that *we save almost 60 hours per month executing reports using Informer.*

When I told my boss, he was absolutely thrilled, recognizing that Informer had more than paid for itself. Informer has an excellent return on investment. You have a great product and a loyal customer!

For more information, including customer-driven demonstrations of Informer software, visit Entrinsik at www.entrinsik.com or call 888-703-0016. ■

Leading Analyst Firm Places Datatel in Leaders Quadrant of "Magic Quadrant" Report

Datatel, Inc. announced that Gartner, Inc. has positioned the company in the Leaders Quadrant of its Higher Education Administrative Suites Magic Quadrant for 2008. This marks the eighth consecutive time Gartner has acknowledged Datatel with this placement.

According to the Gartner, Inc. report, vendors in the Leaders Quadrant have the highest combined measures of ability to execute and completeness of vision. They are performing well, are prepared for the future, and have strong customer bases that have fully implemented their suites. Additionally, each of the vendors in this space has functionality beyond the minimum requirements for inclusion in this Magic Quadrant. These vendors' assets include strong channel partners, consistent financial performance, broad platform support, good customer support, the most-comprehensive and scalable products in the market, and the ability to deliver all the key higher education administrative suite capabilities. Strong vision reflects a solid product and go-to-market strategy for higher education, combined with a management vision for higher education.

"We are pleased once again with Gartner's positioning of Datatel in the Leaders Quadrant," said Jayne Edge, Datatel vice president of strategic planning and marketing. "Their recognition of our strong vision, strategic direction, client commitment, and deep knowledge of higher education is gratifying as we provide the best solutions for higher education institutions to build effective Strategic Academic Enterprises."

The Magic Quadrant is a graphical representation of a marketplace at and for a specific time period. It depicts Gartner's analysis of how certain vendors measure against criteria for that marketplace, as defined by Gartner. Gartner does not endorse any vendor, product, or service depicted in the Magic Quadrant, and does not advise technology users to select only those vendors placed in the Leaders quadrant. The Magic Quadrant is intended solely as a research tool, and is not meant to be a specific guide to action. Gartner

disclaims all warranties, express or implied, with respect to this research, including any warranties of merchantability or fitness for a particular purpose.

About Datatel, Inc.

Datatel provides fully-integrated administrative and academic software systems and professional services to colleges and universities, for building Strategic Academic Enterprises in support of student success. Serving five million students at more than 760 institutions throughout North America and overseas, Datatel has exclusively focused on higher education for 30 years. Datatel is headquartered in Fairfax, Va. For more information, visit www.datatel.com. ■

Brian Leach Named An IBM Data Champion

Brian Leach, of Brian Leach Consulting Limited, has been designated an IBM Data Champion Award for 2008-2009. As described by IBM:

> Technical communities, web sites, books, conference speakers and blogs all contribute to the success of IBM's Data Management products. But these activities don't run themselves. Behind the scenes there are dedicated and loyal individuals who put in their own time to run user groups, manage community web sites, speak at conferences, post to forums, and write blogs. Their time is uncompensated.

> IBM is honoring the commitment of these individuals with a special designation — Data Champion — as a way of showing our appreciation for the time and energy these exceptional community members expend.

> Data Champions are objective experts. They have no official obligation to IBM. They simply share their opinions and years of experience with others in the field, and their work contributes greatly to the overall success of Data Management. (intl-spectrum.com/s1010)

A past president of the International U2 User Group, Brian is very active in the IBM U2 community. As a developer, consultant, and writer, he has authored several training guides for UniVerse developers and knowledge-base articles for the U2UG web site. His articles on leading edge technologies are frequently published in *International Spectrum* magazine.

When asked for his reaction to the award, Brian replied, "I'm delighted to receive this award, and even more delighted to see IBM U2 (UniVerse and UniData) represented on that list. We need to keep it there."

For more information about Brian Leach's products, books, and services, visit his web site at http://www.brianleach.co.uk. ■

New Products

BlueFinity Enables Seamless Data Transfer Between MultiValue Databases and SQL Server Using mv.SSIS

BlueFinity International, a member of the Mpower1 Group of Companies, is pleased to add a brand new product to its range of Microsoft development tools for MultiValue users. BlueFinity's mv.SSIS allows MultiValue users to extract data from any major MultiValue database (in both bulk and filtered export mode) into MS SQL Server using Microsoft's SQL Server Integration Services product.

Microsoft SQL Server Integration Services (SSIS) is a sophisticated, graphical-based integration tool which allows data flow processes to be built for purposes such as data mining, data migration, reporting, and OLAP (OnLine Analytical Processing). Once designed, these processes are compiled into 'packages' which can then be distributed and run on any machine with the SSIS tools installed to address complex business needs.

BlueFinity's new mv.SSIS product provides a tightly integrated, natural extension to the native SSIS environment allowing developers to seamlessly incorporate MultiValue data sources into their existing or new SSIS workflow packages. It allows drag-and-drop creation of standard SSIS packages within the SSIS design studio, providing seamless transfer of data between MultiValue databases and SQL Server 2005 and 2008 by utilizing SSIS.

"mv.SSIS effectively opens up the entire SQL Server set of tools and utilities to MultiValue database users," explains David Cooper, Lead Developer at BlueFinity. "With our mv.SSIS product, we have ensured that the way in which MultiValue data is accessed within SSIS is a natural extension to the SSIS way of doing things. This brings many advantages to the user/developer, not least the ability to send e-mail messages in response to SSIS events, update data warehouses, clean and mine data, and manage SQL Server objects and data. By providing a tight, seamless, and natural extension to the standard SSIS environment, mv.SSIS makes SQL Server-based capabilities within your MultiValue environment a reality."

mv.SSIS provides two new objects within the SSIS toolkit and utilizes BlueFinity's mv.NET product to connect to a MultiValue database: mv.NET Connection Manager and mv.NET Data Source.

The Connection Manager provides an SSIS-aware connection to an mv.NET-based data provider which then can be used throughout an SSIS package. Data sources are used within 'Data Flows' and use a Connection Manager to connect to a data provider and then feed data into the rest of the Data Flow.

To connect to a MultiValue database from within an SSIS package, the user must first add an mv.NET Connection Manager to the package then within a 'Data Flow Task' add an mv.NET Data Source. From within the custom mv.SSIS data source editor the user can define what data they want to read into the process. The rest of the transformations and destinations are then defined as per the user's requirements using any of the standard SSIS 'Data Flow Transformations' and 'Data Flow Destinations'.

SSIS provides comprehensive facilities to enable schedules to be configured to manage the data transfers. However, on MultiValue databases that support triggers, transfers can be initiated using mv.NET's ability to monitor the database for significant events. These events, once detected by a .NET application using mv.NET, can then be serviced by the execution of SSIS packages.

"mv.SSIS is another piece in the jigsaw of allowing MultiValue developers to participate as first-class citizens in the Microsoft world," says Cooper. "It's by no means the final piece, but it's an important one and we are certainly experiencing a huge amount of interest in mv.SSIS since its launch."

BlueFinity products provide the developer with the greatest flexibility to ensure that applications can be deployed across the widest range of databases; they are currently available on D3, jBASE, mvBASE, Reality, Power95, QM, UniData, UniVerse, and UniVision and others by request.

BlueFinity is planning a one day technical seminar in Denver, Colorado on Friday, March 27th at the Westin Westminster Hotel to showcase the product set and take an in-depth technical look at the advanced functionality by way of live demonstrations and presentations with BlueFinity developers. More information is available at http://www.bluefinity.com/v4/denver2008.html.

About BlueFinity International

BlueFinity International (www.bluefinity.com), part of the Mpower1 group of companies, supplies leading-edge software development tools and consultancy services to the MultiValue database and Microsoft developer communities. Founded in 2002, BlueFinity has created a series of products. Its flagship product — mv.NET — is a comprehensive solution for developers wishing to access MultiValue databases from within Microsoft's .NET environment. ∎

New Release of MITS Report Expands Database Support, Improves Usability, and Integration Capabilities

MITS, leading provider of reporting and business intelligence solutions, announced a major new release of its operational reporting solution, MITS Report.

Eighteen months in the making, MITS Report 2.0 includes complete integration with a wide range of MultiValue and relational databases, giving customers more consistent and powerful reporting capabilities in today's multi-platform computing environments. The release also constitutes a major step forward in usability, with a new look and dozens of data access, presentation, and integration features.

Enhanced MultiValue and Relational Database Support

With version 2.0, all MITS report functions can be used with nine additional databases. Support has expanded to a combination of MultiValue and relational platforms, including: TigerLogic D3, OpenQM, Northgate Reality, Microsoft SQL Server, MySQL database management system, PostgreSQL, InterSystems Caché, Oracle and Apache Derby. MITS Report continues its complete support for IBM's UniVerse and UniData platforms.

"It is not uncommon for companies to utilize more than one database platform," says Gary Owen, MITS Vice President of Software Operations. "Because MITS Report supports multiple databases, users can create reports while remaining in a single, familiar environment and interface."

Easier Integration, Excel Connectivity, and PDF

The new release also adds a range of new Web application programming interfaces (APIs) which enable software solution vendors to more thoroughly embed MITS Report within their own applications.

"This new, web services-based integration is important to end users as well as software providers," adds Owen. "Now any application, wherever it was developed, can be integrated with MITS Report via web services." While Internet-based applications are a great fit for the new interoperability features, MITS has successfully integrated MITS Report with .NET applications as well.

The new release helps users set up dynamic connections to MITS Report from within Excel workbooks. "Now users can connect to a report from within any Excel document," explains Owen, "similar to the way Excel connects to a database." Once connected, a spreadsheet can be automatically updated as the underlying data presented by the report changes. Excel power users will appreciate the ability to integrate the filtering, summarizing, and formatting capabilities of MITS Report with consolidation, pivot tables, and other advanced Excel features.

Because Excel isn't the only way users share data, MITS has added PDF export capability to MITS Report. End users will enjoy benefits of creating and distributing reports via email directly through the MITS Report interface.

Usability and Reporting Features

Every end user will immediately notice the new look of MITS Report. "Everything just fits together more cleanly," says Owen. "We've also made countless refinements to the user interface, so the solution works more intuitively."

In one subtle but powerful example, clicking on the "last updated" date on a report results in an immediate option to refresh the underlying report data for up to the minute results. In another, users can jump instantly to a specified page instead of scrolling through a report.

Both IT professionals and end users will appreciate the new prompting features, which allow report authors to set up contextual menu-based prompt lists for filtering reports. The feature also allows a single report to replace many in cases where the only difference across reports is, for example, a date range, department identifiers or other easily selected data.

Subtotal and total lines — or rows on summarized reports — can now present averages as well as sums for any column. Summarized reports are easier to use, too, now that users can choose to expand any subtotal into its component details without expanding detail throughout the report.

Even more powerful is the ability to hyperlink a report column to Web content or specific application function. A user checking an inventory report, for example, can see an unusual stock level, click on that column and row, and instantly call up a related vendor or item screen in another application. "Operational reports are meant to be actionable," explains Owen. "This lets people see a report item and instantly click to the relevant part of another application or web site so they can do something about it."

Simplified Maintenance, Improved Performance

MITS included powerful new performance enhancements and tools to assist internal and external support staff in the new release. In the new Support Console, administrators can review commands and notifications for faster, easier problem-solving..

Administrators can now manage report source updates – the tasks in which reporting data are pulled from operational databases – more precisely, thanks to new incremental updating options. They can now also enjoy email notification when report source updates are completed.

"We focused considerable effort on performance improvement," reports Owen, "especially regarding complex and ultra-large datasets."

With thousands of hours invested in the development and testing of this release, there is no doubt that MITS Report 2.0 will satisfy the operational reporting needs of independent software vendors and end users alike.

For more information, visit www.mits.com. ∎

BY MIKE RUANE, REVELATION SOFTWARE

Indexes in OpenInsight

Part 2: Practical Use

I ndexes. There is probably no one thing that can improve the reporting and searching speed of a system as much. Yet to so many users (and even some developers), indexes are magic. It shouldn't be so: indexes are easy to install and relatively easy to understand. And, when applied correctly, can save untold hours of reporting and processing time.

The Three Indexing Types in OpenInsight

OpenInsight supports three different types of indexes: Btree, Cross Reference, and Relational indexes. The processes of adding, deciding when to add, maintaining, and removing each of these types of indexes are covered in the rest of this article.

Adding a Btree Index

Indexes can be added by way of the command line, using the CREATE_INDEX command, or by using the Database Manager. From the Database Manager, the user should choose the Utilities-Indexes-Add menu choices. The user is then presented with the menu choices seen in figure 1.

Choosing the Btree choice brings up the screen seen in figure 2. The Filename list box contains a list of all files in the current application. The Column Information grid is filled in once a file has been chosen from the Filename list.

The build options give two choices of how the index should be built. If the Case Sensitive checkbox is checked, the index is built using whatever the case of the column value being indexed. So, if the column being indexed contains the values 'Smith' and 'SMITH', they are stored as separate entities in the index. Since they are stored as separate values, a search on 'SMITH' will not bring back the values for 'Smith', nor will the reverse. Except for very specific instances, case sensitive indexes are not used, and so this checkbox is usually, and by default, not checked.

The Immediate Build checkbox causes the index to be built immediately. When an index is added to a field in a file, triggers are applied to the file. When the Immediate Build checkbox is checked, then the index is built as soon as the triggers are executed. If the checkbox is not checked, the triggers are loaded but the index is not built until some other action occurs.

Figure 3 shows a Btree index being added to the STATE field in the CUSTOMERS file. The file has been chosen
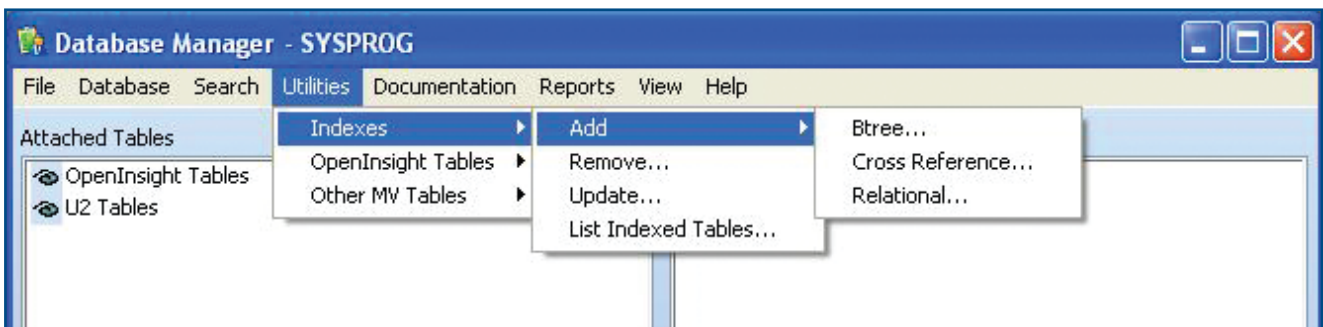

Fig. 1 The Index Add menu choices

from the Filename field, the field has been selected in the Column Information grid, and the Add button has been clicked. The status bar at the bottom of the screen will show progress of the index rebuild. At the completion of the index build process, the status line is updated as seen in figure 3.

An index can also be created programmatically, or from the command line, using the Create_Index command (fig. 4).

The syntax for adding a Btree index to the STATE field in the CUSTOMERS file, as seen above, can be done with the command in figure 5. A Btree index is then created on the Customers file, on the state field, it is case insensitive, and it is built immediately.

When Should A Btree Index Be Added?

It is simple to add a Btree Index, and after reading the first part of this article (in the November/December 2008 issue) you know that the index values are stored in the associated index file in nodes with an associated data value/ key pairing. Indexing takes up disk space and does take some processing time, but not all that much. So, is it a good idea to add a Btree index to all fields? No.

A good rule of thumb to follow for adding Btree or Cross Reference indexes is to add them to fields that are used to select records or define sort order in a process. More simply, add an index whenever you have a process that performs a WITH or BY clause on a field.

Maintaining a Btree Index

In general, Btree Indexes take care of themselves. When data is modified in an
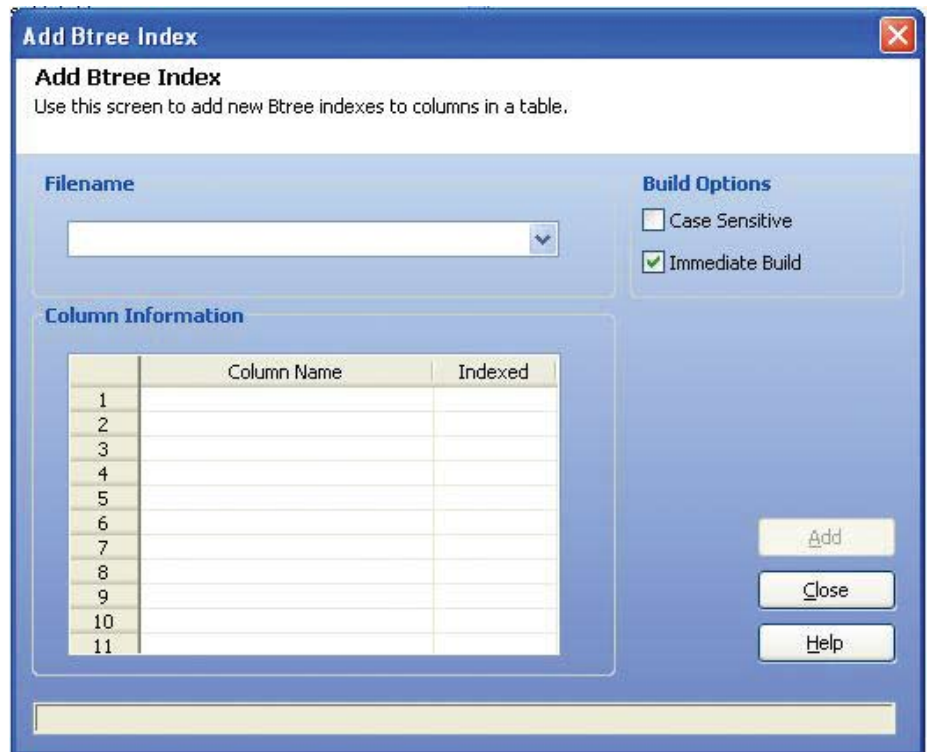
Fig. 2 The Add Btree Index window
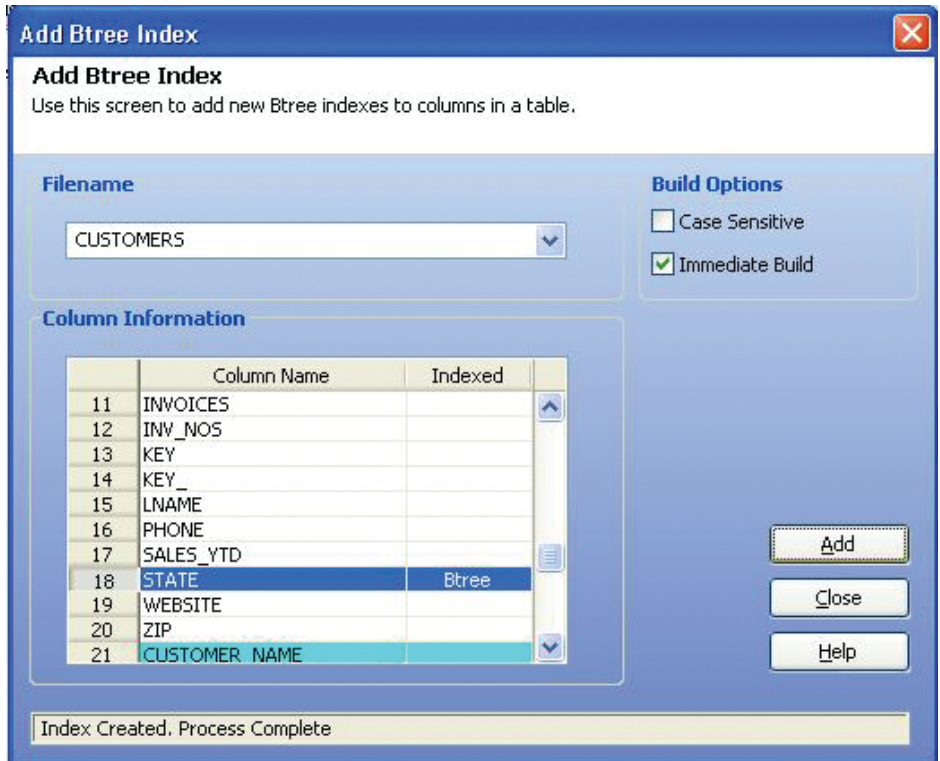
Fig. 3 A Btree index added to the State field of the Customers file

```
Create_Index(Index_Type, Table_Name, Field_Name, Case_Flag, Create_Mode)
```
Fig. 4 Create_Index command syntax

```
Create_Index(1, 'CUSTOMERS', 'STATE', 0, 1)
```
Fig. 5 Syntax for adding a Btree index

INDEXES IN OPENINSIGHT – PART 2: PRACTICAL USE

Continued from page 25

indexed field, the triggers on the field (in the form of SI.MFS) make sure that the information is tracked as changed and put into a pending index transaction record. The only choice left to the developer is how and when these pending index transactions should be processed.

Since a SELECT or LIST statement will use the index to select or sort the records to be processed, it stands to reason that the indexes should be up to date. It also makes sense to have the indexes updated by the system if the system has some idle time. Luckily, both can be accomplished in the OpenInsight Environment settings. The Index settings can be found by opening the Database Manager, choosing Database-Environment Settings, and then clicking on the Indexes/Reports tab. Figure 6 shows the screen.

The items circled in red in the lower right-hand side of Figure 6 are related to how indexes are updated in Open-Insight. The first choice labeled 'Delay Before Indexing' indicates how many

seconds the system should wait before it begins updating pending index transactions. A blank entry or a zero indicates that no updating takes place in a timed manner. Case Sensitive Query is used in conjunction with the Case Sensitive flag being set during an index build.

The 'Update Before Query' checkbox, when checked, indicates that during a SELECT or LIST statement any pending index transactions for fields that are used in selecting or sorting the statement should be updated. Setting this flag to true will ensure that most of the processes will have their indexes as up to date as possible. An exception to this is where the developer uses system routines such as BTREE.EXTRACT or IndexLookup in their programs. These routines don't know whether or not the pending index transactions are up to date, nor do they care. It is up to the developer to use the Update_Index command to update the indexes.

The Update_Index Command

When a developer needs to update indexes manually, they can use a system stored procedure called Update_Index. Update_Index is used to update or re-
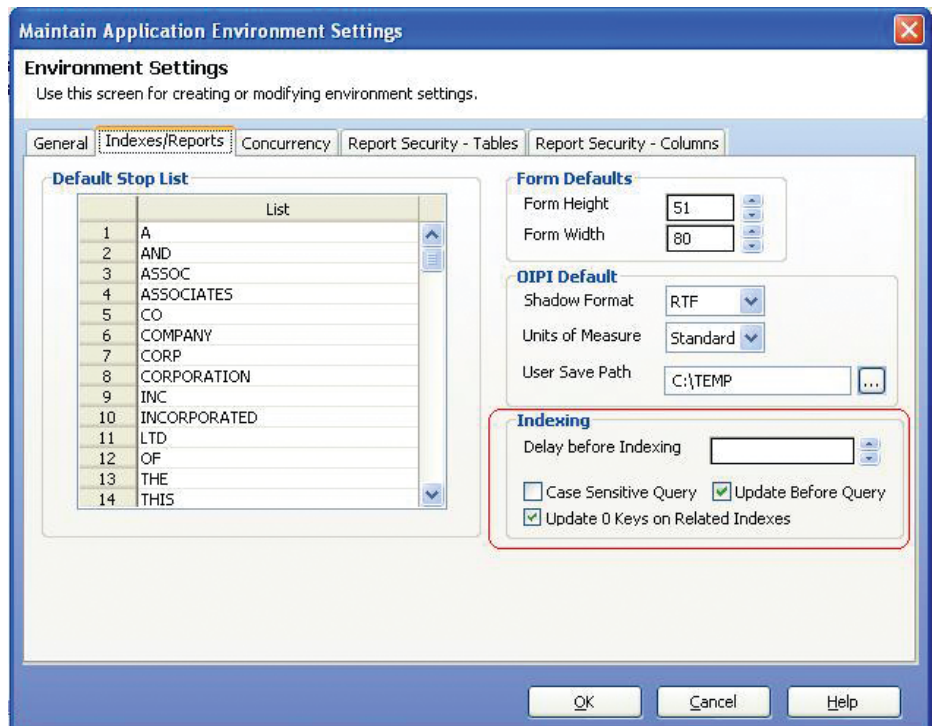

Fig. 6 The Indexes/Reports tab of the Database Manager

build an index. An update looks for pending index transactions and processes them. A rebuild will clear out all of the indexed values and related keys out of the indexing table but retain the control information. It will then rebuild all of the indexes defined for that file. The Syntax is shown in figure 7.

Table_Name is the name of the indexed table, or a null value for all indexed tables. Indexed_Field_Name is the name of an indexed field in the table, or a null value for all indexed fields; and Flag which is set to 0 for update or 1 for rebuild. So, the command (fig. 8) will update all pending index transactions.

Indexes can also be updated or rebuilt using the Database Manager. By opening the Database Manager and choosing Utilities-Indexes-Update, the Update Indexes screen is displayed, as seen in Figure 9.

This screen shows a list of all the indexed fields and files in the system. Note that the Update and Rebuild buttons are still disabled. This is because no rows in the grid have been selected. Selecting one or more rows in the grid will enable the two buttons, and the developer can choose which function they'd like to perform. Figure 10 shows STATE index in the CUSTOMERS table selected, and the buttons enabled.

The All checkbox, when selected will allow the developer to rebuild or update all indexes in the system. In general, it is not a good idea to rebuild all the indexes in a system all at once.

Removing a Btree Index

Removing a Btree index is as easy as adding one. From the Database Manager, choose Utilities-Indexes-Remove. The user is presented with a screen that looks very similar to the Update Indexes screen, except that instead of buttons for updating or rebuilding indexes, there is just one button labeled Remove.

```
Call Update_Index(Table_Name, Indexed_Field_Name, Flag)
```

Fig. 7 Update_Index syntax

```
Call Update_Index('', '', '')
```
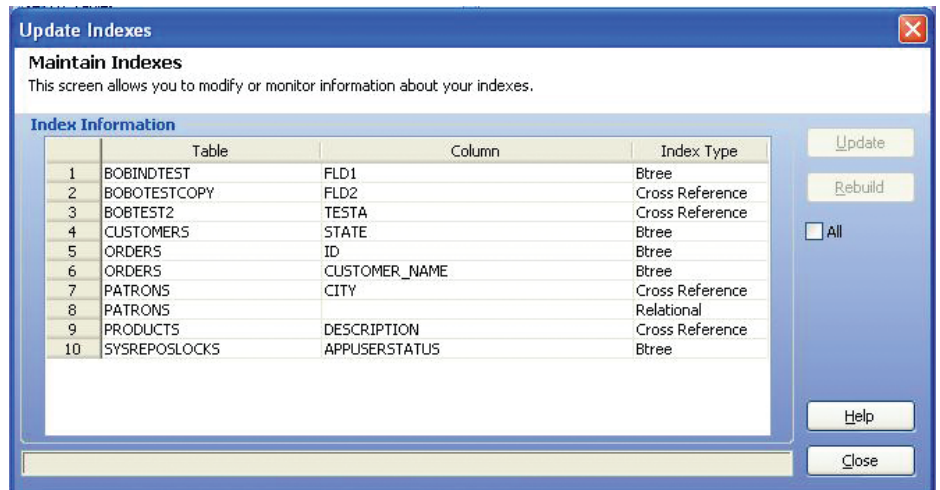
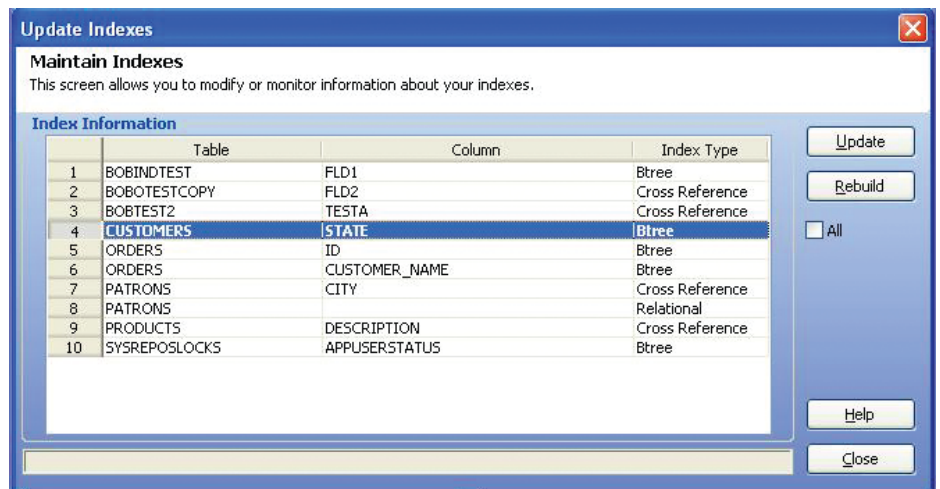**Fig. 8 Update_Index command**

Fig. 9 The Update Indexes screen

Fig. 10 An index selected for updating or rebuilding

BY CLIFTON OLIVER

Improving Existing Software:

Refactoring

Part 2

In the previous issue**,** we made the case for refactoring as an ongoing philosophy and programming ethic. This installment will discuss some of the key indicators that suggest a section of code might be a candidate for refactoring. These are known among refactoring advocates as code *smells.*

Odiferous Code

We have all had the experience of picking up a program, taking one look at it, and saying, "Ewwww! This code stinks!" It's like walking into a house and seeing trash, filth, and molded dirty dishes everywhere combined with a stench that makes your eyes water. You could clean it up, but you really would rather torch it and burn it to the ground. (We have all inherited some programs like that.)

Smells, on the other hand, are something more subtle. For one thing, they don't smack you in the face; sometimes you have to go looking for them. Also, they don't automatically mean the code is broken or even bad. Code smells are more like walking into your house and thinking you smell the faint odor of gas. It's an indicator that something *might* be wrong. You go check the stove to see if a pilot light is out. Then you check the heater. And then you discover that your kid was working on her science fair project earlier and didn't air the house out afterwards. Nothing's wrong, in the repair sense of the word.

To take the analogy one step further, assuming your budding scientist wasn't working on her hydrogen sulfide experiment, you might find something suggesting action. If a pilot light is out, relight it. If a flex pipe is cracked, replace it. Do not put duct tape on it. These sets of appropriate corrective actions are a catalog of refactorings you carry around in your head.

Just like a house may occasionally present a typical smell — gas, mold, dead plants — program code has smells that are typical. Let's explore some of the procedural code smells that we may find in our MultiValue Basic programs.

Long Procedures

Since Structured Programming was first introduced, there have been people who have confused it with GOTO-less and GOSUB-less coding. Have you ever seen a program that had a LOOP statement on page one and its REPEAT statement on page 16? Within it you had ten or more levels of indentation of LOOPs, IFs, and FOR-NEXT structures. That is an extreme example of a Long Procedure. But in the refactoring sense, long procedures can be many lines shorter.

How many lines constitutes a long procedure? That's a matter of opinion. Back in the green-screen days working over dial-up, my rule was that no procedure (GOSUB subroutine) would be over 20 lines. That is what would comfortably fit on the screen to be read and understood without losing concentration going up and re-listing lines. (Remember, back then we didn't have scrolling ability. We had dumb terminals and the ED editor.) I've loosened up on that rule a bit now that I have adjustable font sizes. But I still try to limit a procedure to doing only about five conceptual things.

One refactor for long procedures is to simply start at some level of indentation and put that code into a subroutine, replacing it with a GOSUB in the procedure being modified. See figures 1 and 2 for an example. Repeat with other procedural blocks at the same level of indentation until the procedure being refactored is readable and easier to comprehend.

```
LOOP
      ...
      LOOP
           ...
      UNTIL ... DO
           ...
      REPEAT
      ...
UNTIL ... DO
      ...
REPEAT
```

Fig. 1 Long Procedure: Before

```
        LOOP
            ...
            GOSUB CALC.ORDER.TOTAL
            ...
        UNTIL ... DO
            ...
        REPEAT

        STOP

    CALC.ORDER.TOTAL:

        LOOP
            ...
        UNTIL ... DO
            ...
        REPEAT
        RETURN
```

Fig. 2 Long Procedure: After

Dead Code

Dead Code is usually code that has been commented out, although it can also be code that is present but not used, like a subroutine with no GOSUB to it. Even if it is commented out, for reasons of perception and cognitive issues beyond the scope of this article, it impedes comprehension of the program. This makes the program as a whole harder to understand and thus more error-prone.

Don't comment out code; delete it. In the modern MultiValue world, leaving an audit trial of program change history with comments is just silly. Get a source code control system.

Duplicated Code

Duplicated code frequently indicates a lack of attention, comprehension, or analysis.

Duplicated code frequently indicates a lack of attention, comprehension, or analysis.

Blocks of code that do exactly, or almost exactly, the same thing are suspect. In many cases they can be replaced by a subroutine — maybe with a simple parameter passed. That way when the logic needs to change, there is only one block of code to analyze and alter. Remember, one of the tenets of refactoring is that programs will have

to be modified in the future to adapt to changing business rules.

CASE Statements

CASE statements can be an issue for two primary reasons. First, there seems to be a tendency to just toss on an additional CASE block when adding another condition and code everything it needs within it — even if portions of that code are identical to code sequences in other CASE blocks. This is a special case of Duplicate Code.

> *In the modern MultiValue world, leaving an "audit trial" of program change history with comments is just silly. Get a source code control system.*

The second issue you sometimes find is where the entire BEGIN CASE-END CASE structure with the same conditions and CASE blocks are duplicated in different places in the program. You should consider extracting this to a single subroutine and calling it from different places.

Speculative Generality

This refers to code, hooks, and feature stubs that currently aren't used, but seem to be there *just in case*. In other words, someone has put something in the program just because they think we might want it someday; it is not part of the requirements, but there it sits, accomplishing nothing except requiring others to read and comprehend it in order to figure out that it doesn't matter. Sometimes it even burns machine cycles. Take it out. If you think that you may actually want it someday, put it in a word processing file, document it, and file it in a library of code snippets.

Unnecessary Variables

I was reviewing some code for a client a few years ago and discovered that every time there was a LOOP-REPEAT structure there was an NDX = 0 before the LOOP and an NDX = NDX + 1 just before the REPEAT. The NDX counter, however, was not used anywhere in the loop. When I asked the programmer responsible about this I was told, "Oh, that was the shop standard at the last place I worked. All LOOP statements had to be written that way."

Hm.

Obviously, these unnecessary variables should be removed, though very carefully, given that we do not have scoping in our language. But what concerns me more when I see something like this is the notion that at some point the program was modified, or even originally written, by a rote programmer. This smell acts as an warning to be on the alert for other issues, like reopening files in a subroutine or recalculating constants.

Explanatory Comments

Comments are considered by most programmers to be a *Good Thing*. There is one kind of comment, however, that leaves a bad smell, not because of it's wording, but because of the very reason for its existence. These are comments that try to explain how a piece of code works.

Comments should answer the questions of *why* and *what*. Comments that answer the question *how* are a red flag that a section of code is probably obscure, incomprehensible, or highly dependant on platform side effects. The refactor here is to fix the code.

The Catalog

One of the things you will want to do is to collect a set of documentation about what kind of things in your code should be candidates for refactoring and the shop-standard way in which to refactor them. This is important. Refactoring

IMPROVING EXISTING SOFTWARE: REFACTORING — PART 2

is an ongoing process, not a one-time project. It will be done by everybody in your shop, including those people you haven't hired yet. You want everybody doing it the same way.

Your catalog entry for a particular refactor also should contain warnings about potential problems.

Figure 3 shows a FOR-NEXT loop that recalculates its ending value in the FOR statement. Refactoring this as shown in figure 4 is appropriate. But the catalog for this refactor pattern should warn not to take the "next step" (fig. 5). While to an experienced MultiValue programmer the off-by-one error introduced is well known, a less-experienced programmer may not think of the different behavior between the COUNT version versus the DCOUNT version when the dynamic array is empty. Warn them so they stick to the pattern documented.

```
FOR I = 1 TO COUNT(EXT.PRICES, VM)+1
      ...
NEXT I
```

Fig. 3 Recalculating Constant: Before

```
EXT.PRICES.LAST = COUNT(EXT.PRICES, VM)+1

FOR I = 1 TO EXT.PRICES.LAST
      ...
NEXT I
```

Fig. 4 Recalculating Constant: After

```
EXT.PRICES.LAST = DCOUNT(EXT.PRICES, VM)

FOR I = 1 TO EXT.PRICES.LAST
      ...
NEXT I
```

Fig. 5 Recalculating Constant: Bug Introduced

When to Refactor

How often do you refactor your code? Constantly. Remember, refactoring is not a project; it is an ongoing commitment to improving the quality of your software, a little bit at a time. You do *not* say, "Well, I think I'm going to have these two programmers spend the next two weeks refactoring the A/R module." The success of refactoring comes from making *small* changes to code over time.

Any time you touch a program, you look for just one small thing you can improve in addition to the reason you are modifying the program in the first place. Fix a bug; refactor a 30 line long procedure. Add a new pricing code table lookup; refactor a single FOR-NEXT loop that recalculates its ending value in the FOR statement. Look up the definition of the term *Kaizen*.

The Risk

Nobody is going to be nuts enough to claim there is no risk involved when you refactor something. After all, you are changing the program. But you are already changing the program anyway, so you are going to be testing it anyway, one would hope.

Now, if the idea of refactoring still makes you nervous, I would suggest you should be nervous about fixing bugs and changing or adding features. It sounds like you might not have a lot of confidence in your testing and QA procedures.

But that is the topic of another article. **IS**

CLIFTON OLIVER is an independent consultant with 35 years experience in MultiValue systems and databases. His expertise includes application development, performance tuning, system conversions, project management, and training.

Clifton Oliver & Associates (619) 460-5678; wco@oliver.com

INDEXES IN OPENINSIGHT – PART 2: PRACTICAL USE

Using the same behavior as the Update Indexes screen, the button is disabled until an indexed field is selected. Figure 11 shows the Btree Index on the STATE field in the CUSTOMERS table being removed.

There is also a system stored procedure for deleting indexes called Delete_Index (fig. 12). Table_Name is the name of an indexed table or is left null for all indexed tables; where Indexed_Field_Name is the name of an indexed field in the table or null for all indexes; and Index_Type is the type of Index. So, to remove the STATE field's index in the CUSTOMERS table, the command is as shown in figure 13.

Adding a Cross Reference Index

A Cross Reference index acts much like a Btree index, but has an intermediate step. Cross Reference Indexes are usually used on fields that contain a string, rather than a single value. During the index build process the string is brought into memory, is broken up into its component pieces, words such as *A*, *the*, *is* etc are thrown away, and the remaining component pieces are indexed. The throw away words are listed in the OpenInsight environment settings as a Stop List and can be configured by the user.

Because there are so many more options when setting up a Cross Reference index, there are many more options when adding the index. The Add Cross Reference Index screen can be found by opening OpenInsight's Database Manager, and choosing the menu choices Utilities-Indexing-Add-Cross Reference, as seen in Figure 1. A sample of the Add Cross Reference Index can be seen in figure 14.

The Filename, Column Information, and Build Options for adding a Cross Reference index are the same as adding a Btree index. The Delimiters, Stop List, and List are different.

As described above, in a Cross Reference index a string is broken up into component parts and then the parts are indexed. But how is the string parsed
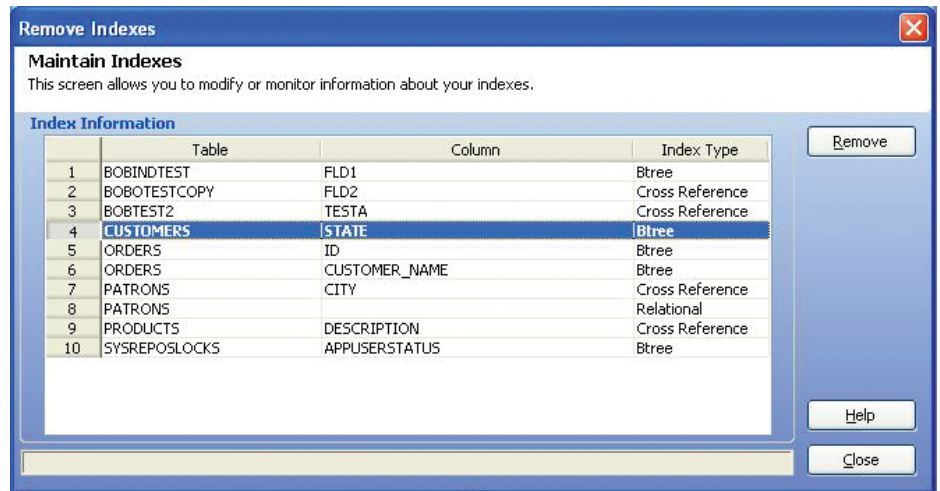
Fig. 11 Removing a Btree Index

```
Delete_Index(Table_Name, Indexed_Field_Name, Index_Type)
```
Fig. 12 Delete_Index syntax

```
Delete_Index("CUSTOMERS", "STATE", 1)
```
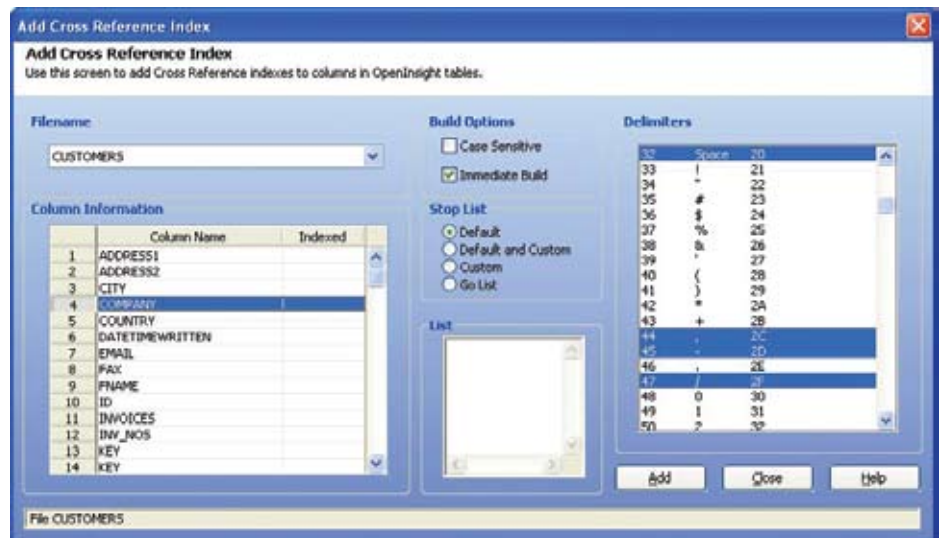**Fig. 13 Delete_Index command**

Fig. 14 The Add Cross Reference Index screen

into pieces? By using the characters specified in the Delimiters list. As a default, the system offers up the space, comma, dash, slash, and backslash characters as delimiters. The string being indexed is parsed by the characters specified in this list. The user can add or remove characters from the list as needed.

The Stop List is a list of words that will not be included in the stored index values. Basically, these are throwaway words and should not be included. Figure 6 shows the default Stop List. In the Stop List group, the radio button determines how the Stop List should be used. Choosing the value for default uses the words in the default Stop List (which can be modified by the user) are excluded from the stored index values. When the value Default and

Custom is selected, the system uses the Default Stop List and then whatever values the user enters into the List box. When the Custom value is selected, only the words entered by the user into the List box at the bottom of the screen are used. Finally, if the Go List value is selected, only the values entered into the List box are indexed.

If using the Create_Index stored procedure to create a Cross Reference index, the syntax is the same as it is for creating a Btree index, but there are two additional parameters for the Delimiter List and the Stop List. To use a space and value mark as the delimiters, and to use the default stop list, use the command shown in figure 15.

Using, Maintaining and Removing Cross Reference Indexes

Luckily, since Cross Reference indexes are the same as Btree indexes on a calculated column, the rules for using them are the same. Add a Cross Reference index on any field where the user will be searching on a part of the contents field, and not the entire contents of the field. For example, a user will probably not want to search on 'The Very Fine Juice Company of America', but they may search on 'juice'.

Maintaining Cross Reference indexes is the same as maintaining Btree Indexes as well, either the using the Index screens or the command lines. Removing a Cross Reference Index is the same as well, except that the Index_Type value is 2 (see example in figure 16.)

```
Create_Index('CUSTOMERS', 'COMPANY', 0, 1, 'SPACE':@fm:'VM', 1)
```
Fig. 15 Create_Index command using a space and value mark as delimiters

```
Delete_Index('CUSTOMERS', 'COMPANY', 2)
```
Fig. 16 Remove a Cross Reference Index command example

INDEXES IN OPENINSIGHT – PART 2: PRACTICAL USE

Relational Indexes

The final type of index in OpenInsight is called a Relational Index. A Relational Index will store keys from records in one table as multivalues in a field in another table. It then maintains this relationship automatically and enforces referential integrity.

As you probably expect, adding a Relational index is different from the other two types of indexing in OpenInsight. Figure 17 shows the Add Relational index screen.

The 'Index From' fields indicate the source of the indexed data: which field from which file needs to be stored. The 'Index To' fields indicate where the data will be stored: in which field in which file. The sort options determine how the data will be stored in the 'Index To' field.

Clicking on the dropdown list for the 'Index From' Filename will give a list of all files in the system. Once a file has been chosen, the Column Name list is populated with the columns found in that file. Clicking on the dropdown list for the 'Index To' Filename will list all files in the system. Clicking on the Column Name will list all the columns in the chosen file. When choosing the columns that the indexed values will be stored in, make sure that you have chose a MultiValue column. If you don't the data will still be stored correctly, but it will not display correctly in a report.

The Sort Options for a Relational Index determines how the indexed values' keys are stored in the 'Index To' Column Name. Append to the Beginning or End is pretty self explanatory

and shows a LIFO or FIFO stack in the field. Ascending Left or Descending Left is mostly used for sorting alphabetic values, while Ascending Right or Descending Right is used for sorting numeric values. Select the appropriate value for your application.

Figure 18 shows how the Courses and Students relational index described in the first part of this article is set up.

Relational indexes can also be created using the Create_Index command (fig. 19). So, to create the same index as

shown in figure 18, the syntax is as seen in figure 20.

As always, check the product documentation for details on the command.

Using, Maintaining and Removing Cross Reference Indexes

Relational indexes are even easier to maintain than Btree or Cross Reference indexes. When a change is made to a related ('Index From') file, the relational index stores the data in the target ('Index To') file immediately, so there are never any pending transactions to

Fig. 17 The Add Relational Index screen
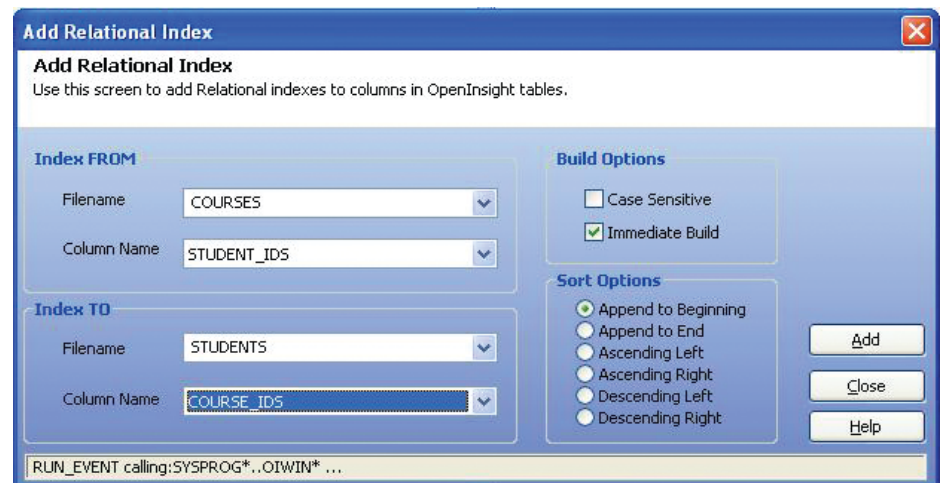

Fig. 18 The Courses and Students Relational Index Setup

```
Create_Index(3 tablename, columnname, casemode, createmode, desttable, destcolumn,
→          sortmode)
```

Fig. 19 Syntax to create a relational index with the Create_Index command

```
Create_Index(3 'COURSES', 'STUDENT_IDS', 0, 1, 'STUDENTS', 'COURSE_IDS', 'TOP')
```

Fig. 20 Example of creating a relational index

INDEXES IN OPENINSIGHT – PART 2: PRACTICAL USE

update, except in the rare case when the target record is locked.

Removing Relational indexes is done the same way as removing a Btree or Cross Reference indexes, either through the Database Manager or a command line. When using the system stored procedure from a program or command line, the Index_Type value for a Relational Index is 3.

Relational Indexes are probably the most powerful indexing construct in OpenInsight. Because the related data is updated automatically, it is easy to create auto-updating tables that work quickly and are always up to date. Watch for a future article on the advanced use of Relational Indexes in OpenInsight.

Theory Versus The Real World: In Conclusion…

In theory, indexing in OpenInsight improves selection and reporting speeds, makes searching easier for the user, and maintains related data for us automagically. In the real world, this holds true as well. OpenInsight's indexing works well for years without any user intervention. It is an almost pain free way to improve the performance of a system. If your OpenInsight system does not make full use of indexing, it should. Review these two articles, review your system, and make changes where necessary. You'll be glad you did. **IS**

MIKE RUANE is President and CEO of Revelation Software. He has authored many articles on OpenInsight, given presentations at various conferences and user groups, and still supports clients using the Revelation Product suite. He can be reached via email at mike@revelation.com.

ERP SYSTEMS AND THE MULTI-VALUE ADVANTAGE

ten a department has a custom written application that exactly follows their work practices.

The departments may have to change their work practices to follow best practices, and there may be a lot of resistance, especially if the way they do their work is perceived as providing a competitive advantage.

Departments have learned how to get something done without waiting for other departments and functions to be completed. In the case of credit approval, for example, if the system won't allow an order to print until the credit department has approved it, and the credit manager is unavailable, it may not be possible to get the order shipped today like the customer service person promised the customer.

Personnel must be trained not only on their own jobs, but about how their jobs interface with other departments. Employees that are still responsible for the day-to-day work may not be able to find the time or concentration to learn a new system. The attitude may be, I'll worry about it when it gets here.

Advantages of MultiValue ERP Packages

MultiValue ERP software vendors have been implementing and providing successful packages for more than two decades. They are experienced and fully understand the interface between departments. The software is easy and quick to modify.

There is much debate about how much to customize ERP software. It is said that it can double the cost of the package. If, after discovery and establishing a thorough and professional implementation schedule, it is determined that certain functions can't be done within the standard best practices, most MultiValue providers can easily and inexpensively customize modules. Some of the big ERP packages are not designed to support customization at all.

If in the middle of an implementation it is suddenly realized that a critical piece of information is not being captured, it is easy to add that field to the package. In addition, the ability to create ad hoc reports—by users, in their own departments—gives back some control to the departments.

We will still have to manage the users' resistance to change, but with MultiValue we can make the change as painless as possible. **IS**

CANDI HART has been an independent consultant in Southern CA since 1980. She was known as Candi Piech when she served as president of CDBMA. She may be contacted at candihart@sbcglobal.net

Tech Tips

B Y   S T E V E   T R I M B L E

OpenQM Read/Write Windows/Linux Directory

OpenQM has made it pretty darn simple to access its underlying operating system.

In the BASIC programming language:

```
OPENPATH pathname {READONLY} TO file.var {ON ERR
statement(s)} {THEN statements()} {ELSE statement(s)}
```

You can instantly find 'your path' by:

```
CURRENT.PATH = @PATH
```

Example:

```
OPENPATH CURRENT.PATH TO CURPATH ELSE
    CRT \Can't open \:CURRENT.PATH
END
```

Note: in BASIC the @DS result provides the underlying operating system delimiter.

```
DIR.LU = @PATH:@DS:"temp":@DS
```

Windows result might be: ACCOUNTNAME\TEMP\

The Linux result might be:

ACCOUNTNAME/temp/ (Linux being case sensitive)

```
EXECUTE "SSELECT ":DIR.LU CAPTURING RESULT RETURNING ERRINFO
```

At TCL: (Windows examples) (Linux examples use the '/')

```
ED PATH:\ACCOUNTNAME\TEMP TEST.TXT
LIST PATH:\ACCOUNTNAME\TEMP
SSELECT PATH:\ACCOUNTNAME\TEMP
```

So, no worries about qpointers or anything else. A slight deviation of syntax and away you go. **IS**

STEVE TRIMBLE of Computerized Data Management, Inc. can be reached at 501-803-0974 or via e-mail at cdmi@swbell.net.

Do you have a Tech Tip to share? E-mail it to editor@intl-spectrum.com

## BUSINESS TECH: BECOMING A SOFTWARE VENDOR - PART 1

Continued from page 9

Tomorrowland

Fast-forward. You have a viable break-even chart. Based on the facts you've assembled, you've taken on investors and partners. Using the capital from the investors, you have developed marketing materials, placed ads, staffed a support center, and finished converting your project into a product. Now you have to look at the point where expenses have to start being absorbed by the revenue. You have to actually make sales.

Steve Alexander wrote an excellent series of articles on this topic, and I won't attempt to repeat it or better it. You can find them in the International Spectrum Magazine archives. (*Editor's note*: see the March/April 2007 issue.) Even if you have hired sales staff, as the President, or CTO, or whatever role you've taken, you are much better off learning the rules of sales.

Fast-forward to the worst case scenario. Oddly, no sales is not the worst case. Low sales is the kiss of death. With no sales, you have failed. With low sales, you have the financial responsibilities of paying staff to support the few customers you have garnered. It is the worst of both worlds.

If you aren't depressed, terrified, or disheartened by now, the next installment will be more optimistic. We're going to talk about why no one will buy your product. **IS**
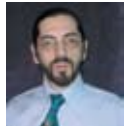
CHARLES BAROUCH is the CTO for Key Ally, Inc. He is also a Past President, as well as currently a Vice President of U2UG, and a regular Spectrum Magazine contributor.

CHARLES BAROUCH Results@KeyAlly.com - *Consulting*; (718) 762-3884x1

Clif Notes
Continued from page 39

I don't blame people for slipping into thinking that majority opinion equals truth. Part of the problem is human nature. Another part might be instilled. Remember when you were first taught to write research papers in school? You had to use at least three sources, correctly format your bibliography, and properly cite your sources in proper MLA style. As much, if not more, emphasis was placed on referring to other authorities as was on the content of the paper. Unfortunately, taking a large number of references as an indicator of truth can slip into normal thinking.

"There aren't any jobs available in MultiValue," (because all the other programmers in my shop say there aren't, so it must be true).

"You cannot make as much working in MultiValue as you can in Oracle," (that's what every MultiValue programmer tells me, so it must be true).

So, a word of caution. Don't be so quick to believe what everybody else says. Investigate. Who said that? Where did you read that? Are they actual numbers that can be checked, or was it a survey of people's opinions?

And one final thing — be skeptical when someone refers to non-attributed studies. Because of the prevalence of believing that references equals truth, in an effort to try to avoid having one's opinion immediately discounted, some people will toss out the general, "Well, after all, studies have shown that blah, blah, blah." It is up to you to ask, "What studies? Where can I find the details on that?"

For example, take the study mentioned in the first paragraph of this column. Bogus. I made it up. Don't feel bad if you didn't pick up on that. It had numbers in it, must be true. Then it's referred to several times in the article (the old self-referential trick). And it is pure, on-the-fly fiction.

But that's ok. You can quote it, if you want to. **IS**

In A Recent Study

BY CLIFTON OLIVER

Clif Notes
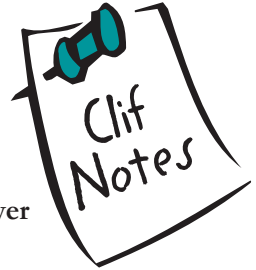
Studies have shown that about 87 percent of people who quote studies have either never read the original study or had a reading comprehension of less than 50 percent.

Those statistics are rather troubling. First, almost everybody references studies in support of whatever position they happen to be arguing. That's not a bad thing. Opinions based on facts are much more compelling, at least to me, than someone who refuses to reason, or reasons in a vacuum. Some people, after being given a statement of fact, shake their head and announce, "I don't believe that," and proceed to elaborate their opinion. Rational folks quickly write these people off as idiots. So in order to avoid looking like an idiot, rational folks frequently offer the facts up front when stating their conclusions. But there are several problems.

First is that, as we see from the study, most of those people are either quoting something they haven't read, or they didn't really understand it when they did read it. It could be a simple matter of the study using specialized terminology, or the topic requires a large amount of prerequisite knowledge in the field. I don't read journals about esoteric physics; I don't have the education in that subject. I wouldn't know a chronosynclastic infundibulator if it particle-ized me. Likewise, I would not expect a historian to understand a computer article filled with OOP techniques.

As to citing a study one hasn't read, well, who has the time? So instead of reading the actual study, we read summaries of and opinions about studies. A lot of times we don't even remember where we have read or heard the mention of the report of the summary of the study. Good luck trying to get back to the original study. Here's a classic example.

Many of you have heard about the "pink monkey." According to the story, some behavioral psychologists were studying the habits and interactions of a tribe of regular, ol' brown monkeys. Alphas and betas, dominance and subservience, aggression and submission, beetles or bananas—the typical sort of stuff social scientists get a kick out of studying. Then, for some reason, they took one of these poor critters, dyed it bright, shocking pink, and put it back in the enclosure with the rest of the tribe. The brown monkeys went bana…, er, nuts and attacked the pink monkey, just because it was dif-

ferent. The researchers had to rescue the pink monkey before it was torn limb from limb. The point being that when confronted by a tribe — whether the tribal identification is genetic, geographic, or choice of football team — being seen as "different" is dangerous.

This study is so well known that there are web sites that are themed on it and don't even explain the term. There is just one problem.

It looks like it's not true. It doesn't appear that there ever was such a study. Some have speculated that it originated with Robert Heinlein in his story *Assignment In Eternity*. But if you tried to support an opinion or win an argument by stating, "Well, in a fictional work by Heinlein, he proved…" you would be laughed out of the conference.

As a side note, the sweet-potato washing monkey study referred to as "The Hundredth Monkey" has also been debunked. Sorry all you personal growth gurus.

It might be tempting to leap to the conclusion that one should never trust any study involving monkeys. But that is not the point. What this demonstrates is that the more a study or authority is referenced, the more it is believed.

Unfortunately, this tendency to believe things just because enough people have sworn it's true has invaded even respectable sources of factual information. For example, take Wikipedia—although many would contest calling Wikipedia a respectable source.

In the November/December 2008 issue of MIT's Technology Review, Simson Garfinkle wrote an article titled, *Wikipedia and the Meaning of Truth* (intl-spectrum.com/s1011). In it he explores the effect Wikipedia and its policies have had on the notion of Truth. As he points out, "On Wikipedia, objective truth isn't all that important, actually. What makes a fact or statement fit for inclusion is that it appeared in some other publication…" If it is published somewhere else, it must be true.

Create a Direct Line between Your Data and Critical Business Decisions

Having rapid access to your data has never been more important. MITS Solutions offer a simple and affordable means to extend the value of your existing system and provide the user-friendly tools you need to make better business decisions throughout your organization. Why wait to employ the information necessary to react and respond to changing market conditions with confidence?

The MITS product line contains two unique but complimentary web-based reporting solutions: The MITS Discover OLAP Business Intelligence System adds powerful data access and advanced analysis functionality to your existing MultiValue system and MITS Report, an ad hoc operational reporting solution optimized for sorting, filtering and summarizing data.

mits
Advanced Reporting
and Business Intelligence

Visit MITS today at www.mits.com or contact us at 888 700 6487