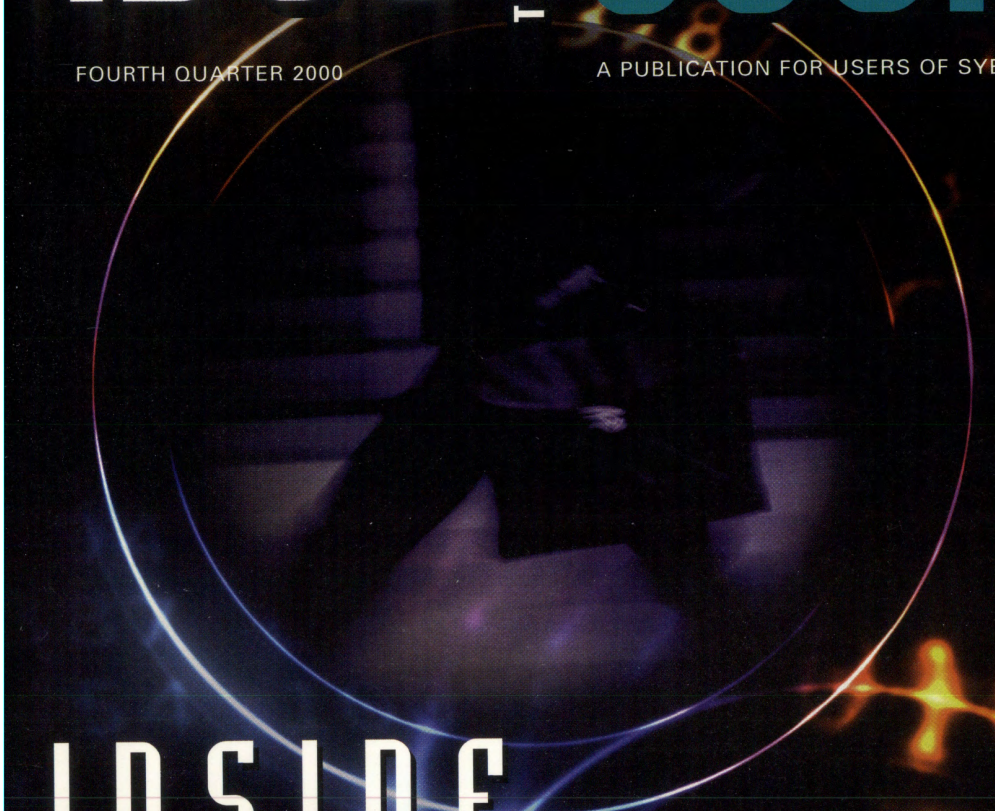
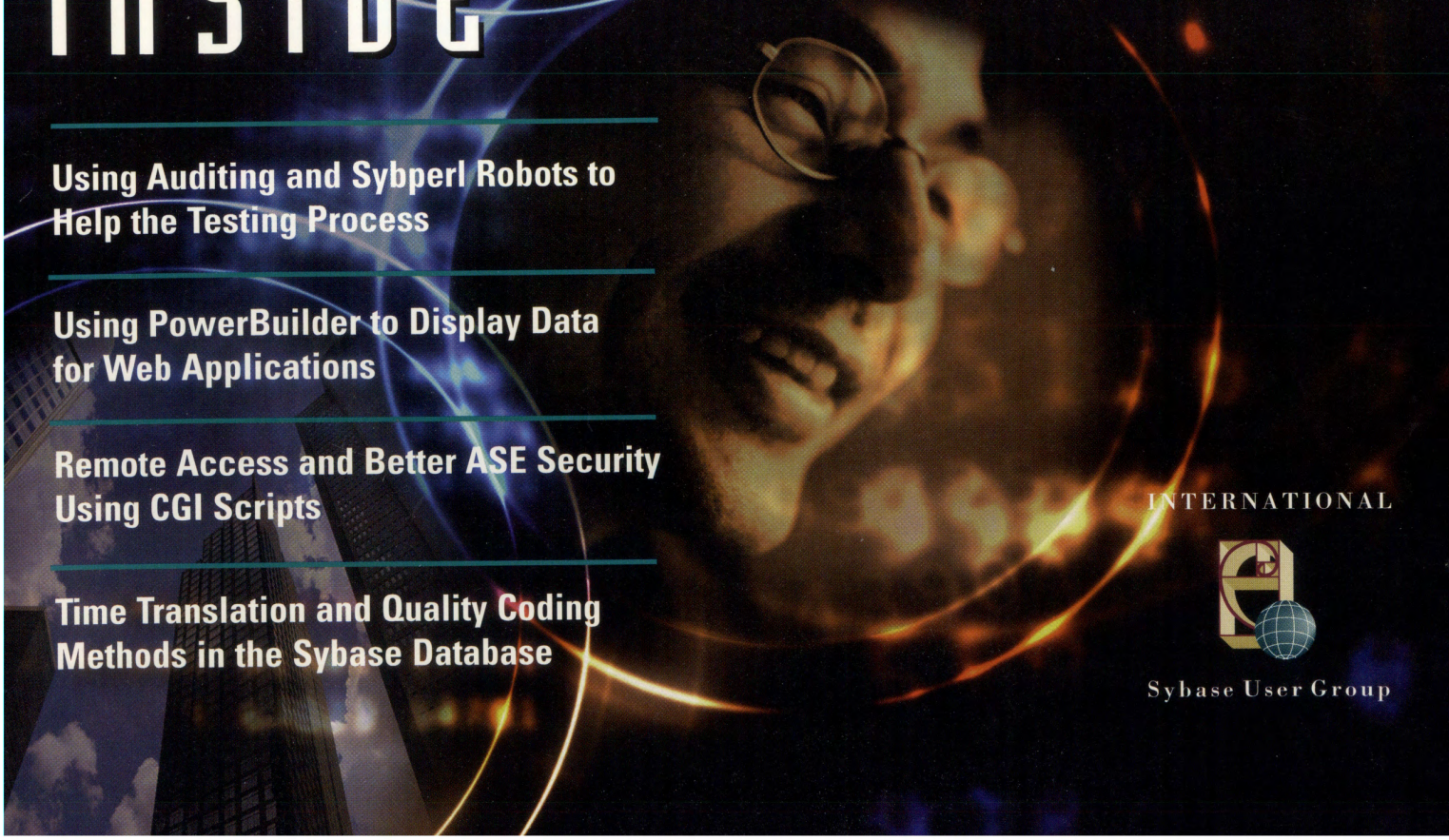
ISUG TECHNICAL JOURNAL

FOURTH QUARTER 2000

A PUBLICATION FOR USERS OF SYBASE, INC. PRODUCTS AND SERVICES



INSIDE



INTERNATIONAL

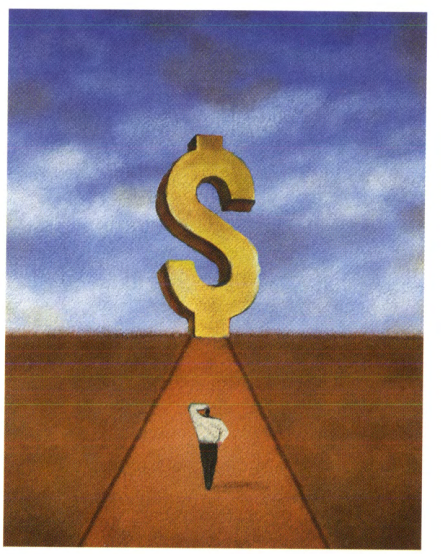
Sybase User Group

PRESIDENT'S MESSAGE

Dear ISUG Members,

Normally I use this column to give you a brief message on how things are going in the Sybase community. This quarter, though, I have some alarming news to share with you. The International Sybase User Group relies very heavily on two key factors: 1) its membership and dues, and 2) the generosity and goodwill of Sybase. Our main revenue generation event in past years has been the North American Sybase conference.

Due to changes in the conference administration at Sybase, however, the mechanisms for providing this influx in membership have changed dramatically over the past two years. As a result of this change, this year we were only able to generate 25% of our anticipated revenues, which we had already drastically reduced due to the aforementioned changes. ISUG is still here, and plans to be here as long as Sybase continues to have users. However, once again I'm calling upon all of you to do everything in your power to get more of your friends and colleagues to join. If you attend a local user group, please talk about ISUG and the benefits we have to offer. If you do not belong to a local user group, seek one out in your area and talk to people about joining ISUG.



On a lighter note, it was my personal honor to present four awards at this year's Sybase TechWave conference. Two of these awards were for the ISUG Outstanding Achievement Award. I feel extremely confident that the recipients of this year's awards (as those in the past) exemplify the best characteristics of those supporting and furthering the knowledge and understanding of the Sybase user community. Again, I would like to extend my heartiest congragulations to all of the award winners this year. See page 30 for more details.

By the time you receive this issue of the *ISUG Technical Journal*, we will be having our fourth quarter meeting at the Sybase Corporate offices. Please feel free to contact your Regional User Group Director (Cindy Bean, Anthony Mandic, or Dorus Kruse), the Conference Director (Cynthia Gill), the Membership Director (Joe Burger), or any other member of the board and let us know what you want for benefits and what can be done to improve the conference experience for you.

Several members approached us during the conference saying that they were very pleased with the large selection of available topics. Some also expressed concern that the time slots for the sessions tended to be too long, and should be 45 minutes to one hour, so that you have an opportunity to attend many more sessions during the conference. We will be presenting these concerns, along with other feedback, to the conference coordinators at Sybase.

The ISUG website has been enhanced and we are now able to take membership registration and renewals online. We have also expanded our credit card services to include American Express. During the conference, we had a number of folks sign up for membership, or even renew on the spot.

Finally, in September I received news that has given me cause for concern. Sybase has announced that it will not be holding an Australasian Conference in 2001. As of this writing, Sybase has also not committed to having a conference in Europe. I would like to ask each and every one of you to visit the ISUG website, where you will find a survey about this and other issues, and complete the survey to let us know what you think about this.

All the best,
Thom Lamb
ISUG President

Table of Contents

FEATURES

DEPARTMENTS

READER INFORMATION

ISUG Technical Journal

ISUG Technical Journal Director
Teresa Larson
Vetcentric.com
410.571.6790
tlarson@vetcentric.com

Managing Editor
Mary Freeman, Freeman Communications
510.525.4863
510.528.6958 Fax
Mary_Freeman@compuserve.com

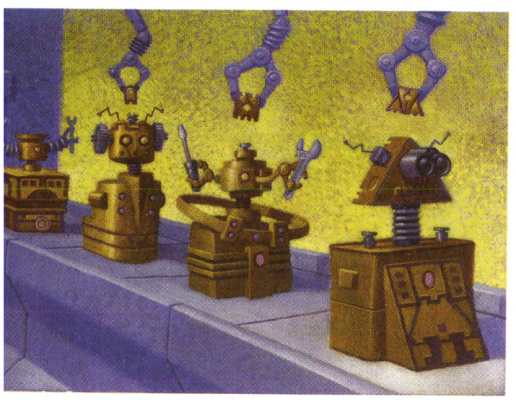
Art Direction
Jerry Jager, JagerCreative
jagercreativ@earthlink.net

Senior Technical Editor
Al Huntley, EDS

Technical Editors
Mark Dirrim, Vantive Corporation
Thomas Lamb, Automated Data Sciences
David Straiton, Sybase, Inc.

Using Auditing and Sybperl Robots to Help the Testing Process

By David Owen



Building reliable testing clients using Sybase auditing capabilities

David Owen is an independent software engineer based in Calgary. He has worked with Sybase ASE & Replication for 10 years and maintains the Sybase FAQ at www.isug.com/Sybase_FAQ. He can be reached at dowen@midsomer.org.

Following up on previous articles published in this journal on the use of Sybperl (the CT/DB Library extensions to perl written by Michael Peppler, *ISUG Technical Journal*, 4th Quarter, 1998), I thought that I would introduce a new testing application that I have written and go through some of the thought processes behind its creation.

I was recently asked by a client to help with separating their Sybase single-database application to create separate environments for OLTP and DSS. One of the potential stumbling blocks was going to be whether replication could cope with the amount of traffic being generated. We needed a means of testing the new configuration with a reasonably representative load. We did have some test management software available to us, but no testing clients had actually been built that worked at all well.

Building testing clients is actually quite a task. After much grinding of mental gears, it occurred to us that we could use production itself. If we could turn on auditing and extract the SQL sent by the clients into files, we could replay those files against our new test environment and produce as realistic a test as possible.

Sybase Auditing

Before we go any further, let's review Sybase auditing and what it has to offer. To get the most out of this article, it would be well worth spending ten minutes to familiarize yourself with auditing and the structure of the auditing tables.

The Sybase auditing sub-system is a very flexible means of capturing information about what ASE is doing. Many different events can be audited, ranging from login successes to recording every change that happens to the data within a specific table. The *Security Administration Guide* contains a lot of information, including a very good section on how to install it. However, the Guide seems to be lacking in detail on how to actually use it. Rob Verschoor's *ASE Quick Reference Supplement* (available at www.euronet.nl/~syp_rob/download.html) contains a better list of the available options.

All of the auditing data is stored in a series of tables within the *sybsecurity* database. The tables are named *sysaudits_01* through *sysaudits_08*. It is recommended that you use at least two tables. Personally, I like to use seven tables. By switching auditing tables at midnight, each table contains all of the auditing information for a particular day in a single table. Switching auditing tables is as easy as putting:

```
sp_configure
"current audit table", 0, "with truncate"
go
```

into the nightly backup script. That way you keep seven days of information for disaster analysis, each day in its own audit table.

Whether you keep auditing switched on all of the time or not comes down to the load on the server and the response

time that the users expect. From personal experience, simply auditing the SQL has a minimal effect, certainly much less than running **sp_sysmon**!

If you decide not to enable auditing for everyday use, and you wish to build the robots described in this article, you need to select a representative day and agree with your users to have auditing on for that day. They will be convinced when you explain to them that the development process will be much improved and they will consequently suffer much less downtime!

Just in case you have not downloaded Rob's book yet, I will tell you that the option we need to enable is **cmdtext**, which will record all of the SQL sent to the server by the clients. The SQL text is entered into the auditing tables *exactly* as it is sent to the server. Sybase does not clean the data by removing any embedded characters such as carriage returns or linefeeds. This is actually a good thing, as we will see later.

The System Architecture

The architecture is very simple. It includes a tool to extract the SQL text from the audit tables and then put it back together in exactly the same form as it was sent to the server; and a tool to replay the robots in the correct sequence with the appropriate usernames. Both parts are written in perl, but only the first uses Sybperl, so I will concentrate on that.

Let's Make Robots

As with all Sybperl scripts, there is a T-SQL component and some perl to process the returned rows. The query to return the rows from one of the audit tables follows:

```
select extrainfo,
    sequence,
    suid,
    spid,
    eventtime
 from sysaudits_02 a
where a.dbid = db_id('YOURDATABASE')
 and a.event = 92
order by suid,spid,eventtime,sequence
```

As you can see, it is a straightforward **select**. **suid** and **spid** are the normal user ID and process ID of the client. **Eventtime** is the time that the event being audited occurred. The event column stores the type of event being audited. In the case of **cmdtext**, it is 92 (prior to ASE 11.5 it was in fact 107, but we do not reflect that here).

If the auditing system needs to store information about the audited event, then it goes into the **extrainfo** column. Not all events require extra information; login success is a good example of such an event. Obviously, the SQL text does need to be stored somewhere, and this is where it goes.

Sequence is used when the amount of SQL text being sent is greater than 255 bytes. The first row has a sequence of 1, and for each subsequent row the sequence is incremented.

The **order by** clause has been chosen with care. We wish to have individual files containing the SQL corresponding to each process connected to the system. To achieve this, we order **by suid** first, to group together all of the records for a particular user. Within that we **order by spid**. This is necessary to allow for the user connecting twice, either concurrently or otherwise.

Next, we wish to have each SQL statement sent by the user in its correct chronological order. And, finally, we want each SQL batch to be in the correct sequence order. **Eventtime** *should* be the same for each statement submitted in a batch, but it would seem that this is not always the case. However, it does not really matter, since it should at least be chronologically increasing within the same batch, and will certainly be chronologically increasing between batches.

Now we need to send the query to the server and process the results. The following pseudocode will help convey what we are trying to achieve.

```
while(<<rows to be fetched>>){
  if (sequence == 1){
    print <CR>"go"
    print <CR>dataRow
  } else {
    print dataRow
  }
}
```

As we mentioned before, the auditing system does not clean the data going into the extrainfo column, so if we print it out to a file without adding or removing any carriage returns, we will reconstruct the SQL exactly as it was submitted. It turns out to be as simple as that. In fact, it turned out to be so simple I was astonished that nobody had done it before.

Below is a slightly simplified perl script for implementing the above pseudo code. This is only an extract, so we have already connected to the server and assigned the database handle to the variable "$srv".

```
$srv->ct_execute(<<EOSQL
select a.extrainfo,
    a.sequence,
    a.suid,
    a.spid,
    a.eventtime
 from sysaudits_02 a
 where dbid = db_id('YOURDATABASE')
  and a.event = 92
 order by a.suid,a.spid,a.eventtime,a.sequence
EOSQL
        );

$firststatement = 1
$last_suid    = -1;
$last_spid    = -1;

while(($rc = $srv->ct_results($restype))
            == CS_SUCCEED) {

 next if($restype == CS_CMD_DONE ||
     $restype == CS_CMD_FAIL ||
     $restype == CS_CMD_SUCCEED);

 while(@dat = $srv->ct_fetch) {
  if (defined(@dat)) {

   ($extrainfo,
    $sequence,
    $suid,
    $spid,
    $eventtime) = @dat;

   if ($last_suid != $suid ||
     $last_spid != $spid) {
    $last_suid = $suid;
    $last_spid = $spid;

    $first_statement = 1;

    if (fileno(SQL)) {
     print SQL "\ngo\n";
    }

    open(SQL,
      "> TMW_$suid:$spid.sql");
   }

  if (1 == $sequence) {

 if ($firstStatement) {
     $firstStatement = 0;
     print SQL "$extrainfo";
    } else {
     print SQL "\n$extrainfo";
    }

   } elsif ($sequence > 1) {
    print SQL "$extrainfo";
   }
  }
 }
}
```

Some Explanations

The argument expected by **ct_execute** is a string containing the SQL you wish to execute. If you look at the code, I have not used a string directly, but rather a feature of Perl called a *here* document, modeled after Unix documents from shell programming. What happens is that everything from the initial **<<EOSQL** to the line *starting* with **EOSQL** are submitted to **ct_execute** as if it were a string. Of course, you are not restricted to **EOSQL**—any matching string will do—but I tend to put something meaningful so that the code reads better. Shell programmers like to use "!" as the string. Note that the positioning of the final string is important and is one of the few circumstances where perl is not flexible. The terminating string *must* be at the very start of the line.

I draw your attention to this feature because it has one big advantage over using a string, and that is that you do not need to worry about quoting within the *here* document itself. For instance, the T-SQL function **db_id** takes a string as an argument, which needs to be quoted or else it will be interpreted as a column. You can use either single or double quotes with the function. However, if we had used a regular string for our query, we would have had to remember which we had used at the perl level and used the opposite in the SQL. Quotes within quotes start to become really messy. A *here* document leaves you to deal just with the SQL.

Just for the record, following is another way to do safe quoting within the SQL of perl programs. You can use either single or double quotes, or both. Note that in this method you might need to escape any @ or ! within the SQL.

```
$SQL=qq!select extrainfo, sequence, suid, spid, eventtime,
"dbl_quote_str", 'sgl_quote_str'
 from sysaudits_02
where dbid = db_id('YOURDATABASE')
 and event = 92
order by suid, spid, eventtime, sequence!;
$srv->ct_execute($SQL);
```

Meanwhile, in order to process a batch within CT-lib, you need to use two loops. The outer of the two loops processes queries within the batch and the inner loop processes rows within each batch. If you do not have both loops, you will be hit by the dreaded *results still pending* error the next time you issue a **ct_execute** for this database handle. This mechanism is very flexible, but can lead to one or two interesting problems with the inner loop jumping through hoops to process the different result sets. It is usually a good idea to try and keep the batches simple, so that the perl code can also be simple.

I can never remember off the top of my head the details of the two loops, so whenever I start a new Sybperl program, I go to my last one and copy the **ct_execute** statement and its corresponding loop constructs direct. I then replace the SQL from the old file with the new stuff and the closing braces, and then start to think about how I want to process the data. Lazy? You bet! After all, it is one of Larry Wall's three rules for what makes a good programmer and I try to live up to that as much as possible!

The inner loop returns an array containing the set of columns defined in the **select** statement. I then divide the array into perl variables of the same names as the columns in the **select** statement for easy handling. This could all have been done in a single statement thus:

```
while(($extrainfo, $sequence, $suid,
    $spid, $eventtime) =
            $srv->ct_fetch) {
...
```

This way is possibly a little quicker, but I think that in this particular case, the **while** looks a little busy. The other way also allows for an additional check to ensure that the array returned by the **fetch** is defined. Checking that the array is defined does not guarantee that all of the elements within the array are defined, and Sybperl will return **undef** (undefined) for any rows where the column was NULL for a particular row in the result set. Strictly speaking, we should check that all of the columns are defined on an individual basis.

However, in this case, all of the columns except **extrainfo** are defined as **not null** in the audit tables, so by putting that column first in the **select** statement and having the single check on the whole array means that we are probably OK. If the code was being used in an environment where it was very important that failure of the script was trapped properly, I may well be tempted to be far more defensive and check each column/variable to ensure that it was returned properly. The sort of scripts I have in mind are those where unexpected failure could result in me being paged in the middle of the night!

The processing of the data is now relatively straightforward. We check to see if the **spid** or the **suid** have changed, and if either has, we open a new output file. Notice that we initialized them both to -1, a value that neither can have in reality, thus ensuring that the very first time through the program, we will execute the "change of spid/suid" code.

The function call:

```
if (fileno(SQL)) {
  print SQL "\ngo\n";
}
```

simply tests to see if the specified file handle (**SQL**) is already open. We need to print a final **go** to ensure that the last statement in a file is properly terminated. On the very first time through, there will be no such open file.

Finally, we write the text to the file. We have to deal with two cases:

1. Sequence = 1
2. Sequence > 1

Notice that I've reversed the normal equality check to say:

```
if (1 == $sequence) {...
```

This is preferred by many programmers, since if you accidentally leave out one of the equal signs, it will not be turned into an assignment statement, which is a very common error in perl scripts and is very difficult to track down. So, if it is the first row, we need to write a **go** into the file. I know that an empty batch is perfectly acceptable to ASE, but it seems ugly, if not a little sloppy, to me. So, I have added an extra check so that we only print the **go** if this is not the very first statement in a file. I know that people will argue that this means that there are a lot of redundant checks in the code now, which will slow it down. Believe me, the time taken to process the rows once they are being returned is as nothing compared to the time for Sybase to return the rows in the first place!

The case when **$sequence > 1** is positively trivial. Simply print the SQL text to the file.

Sychronicity?

So, there we have it, or almost. If we run the SQL text generated by the program as written, it will run in one long stream without ever coming up for air! In other words, it is not truly repeating what the original user did.

Our final feature is to add a delay function so the robot

will only submit the SQL after the same delay has elapsed as in the original session. I thought for a while about how to best implement this. It would be possible for the robot control script to implement the timing. This would mean that the control mechanism was going to have to be far more complicated than I wanted. What I really wanted was for each robot to run itself. I decided that it was simplest to add a **waitfor delay ...** statement between every piece of SQL.

The following perl function takes two Sybase dates and returns a string that is suitable for using with the **waitfor delay** statement. As a small optimization, I have added a check so that if there is no difference between the times, the function returns 0 and the delay statement can be safely omitted. I am not fond of returning mixed result types, but I think that this is an acceptable use. You may prefer to do things differently.

```
sub dateDiff {
  my($date1) = shift;
  my($date2) = shift;

  my($hour1, $sec1, $min1, $yday1);
  my($hour2, $sec2, $min2, $yday2);
  my($secsDiff);

  my($hours, $minutes, $seconds);

  my($totalSecs1);
  my($totalSecs2);

  my($secs_in_a_day) = 24 * 60 * 60;

  (undef, undef, undef, $yday1, undef,
   $hour1, $min1, $sec1, undef, undef) =
                                  $date1->crack;

  (undef, undef, undef, $yday2, undef,
   $hour2, $min2, $sec2, undef, undef) =
                                  $date2->crack;

  $totSecs1 = $hour1 * 3600 +
        $min1  * 60  +
        $sec1;

  $totSecs2 = $hour2 * 3600 +
        $min2  * 60  +
        $sec2;

  if ($yday1 == $yday2) {
    $sDiff =
      abs($totSecs2 - $totSecs1);
  } elsif ($yday1 > $yday2) {
    $sDiff = $totSecs1 +
      ($secs_in_a_day - $totSecs2);
  } else {
    $sDiff = $totSecs2 +
      ($secs_in_a_day - $totSecs1);
  }

  $hours = int($secsDiff / 3600);

  $mins =
    int(($sDiff - $hours * 3600) / 60);

  $secs =
    $sDiff - $hours * 3600 - $mins * 60;

  if (0 == $sDiff) {
    return 0;
  }

  return sprintf("%2.2d:%2.2d:%2.2d.000",
          $hours, $mins, $secs);
}
```

Michael Peppler has done a fantastic job with Sybperl and added some extensions for processing dates and money data types. This function makes use of the **$date->crack** function that splits a Sybase datetime column into its constituent parts. This saved me having to do a lot of tiresome string manipulation. Using **undef** in the array assignment from the crack function call simply thows away that component of the result.

Fundamentally, the function calculates the number of seconds past midnight for each time supplied. If the dates are not on the same day, then the seconds from the first date *to* midnight are added to the seconds past of the other date.

This number of seconds is then broken down into the number of hours, minutes and seconds. Finally, a string is returned in the form **hh:mm:ss.000**.

You will notice that the **dateDiff** function ignores milliseconds and assumes that consecutive SQL commands are no more than a day apart. The milliseconds are too fine a granularity to worry about. By the time we come to replay the various robots, we are going to be plus or minus a couple of seconds anyway, and so worrying about milliseconds is pointless. As for the day-apart issue, I thought about how I would want to use these robots and could see no situation where I would want a robot to wait more than 24 hours between commands. Even 24 hours seems a little excessive! However, if I am being shortsighted and your particular situation requires such delays, then I invite you to make the necessary changes. If you post the changes to me I will endeavour to incorporate them.

This function is then added to the code described above, in the section processing the rows where **sequence** equals 1. The **else** clause becomes:

```
...
print SQL "\ngo";
$delay = dateDiff($lastEventTime,
          $eventtime);
if ($delay ne "0") {
 print SQL
 "\nwaitfor delay \"$delay\"\ngo";
}
$lastEventTime = $eventtime;
print SQL "\n$extrainfo";
...
```

I used a string comparison ('ne' as opposed to '==') to compare the value of **$date** with zero. This is because when **dateDiff** returns zero, either comparison operator will work, but when **dateDiff** returns a string, for inclusion in the **waitfor delay** command, the numeric comparison will fail.

Conclusion

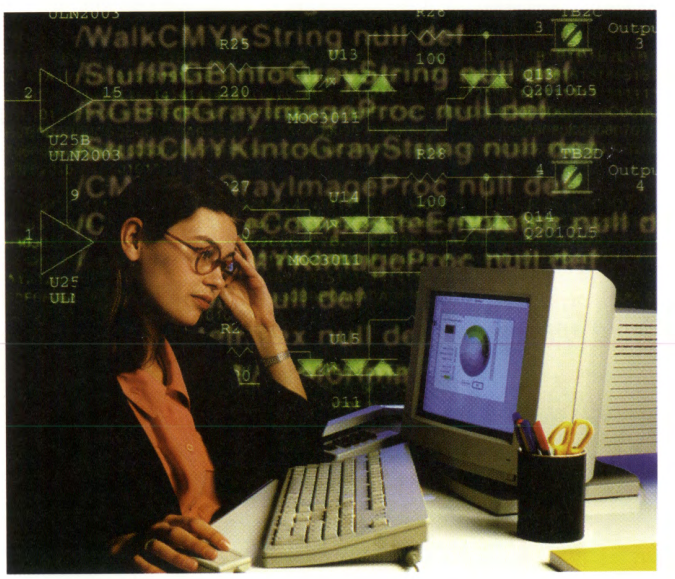
I have not covered the process for playing back the robots, but the complete package is available for download at ftp://ftp.midsomer.org/pub/sybrobots.tgz and contains the full script for generating the robots, the perl script for running them together with full instructions on how to run both scripts and set up your system to provide a really useful set of test clients.

Obviously, it is important that the server you are testing is configured to be as similar as possible to the source server. For instance, if the server that you extract the SQL from has parallel query enabled, then the testing server must also have parallel query enabled. Differences in the hardware are also going to have an impact. It goes without saying that if the source server is a 64 processor E10000 and the destination server is a single processor UltraSparc 5, things are going to run slower on replay.

Mostly, ensuring that the destination server is the same as the source is all about making sure that the results you get back are meaningful. Having parallel query enabled on the source server but not the destination server when you are trying to measure the performance of a new disk configuration will make the test results difficult to interpret in a meaningful way. However, this process is probably fairly useful in trying to ascertain whether or not you could enable parallel query.

At this stage I should issue a warning to potential users. It is important to remember, when using this robot testing process, that it is based on extracting the actual SQL that was sent to the source server and replaying it back to the destination server, preserving the original timing. This may not, however, lead to a perfect replay of the original session.

For instance, in the original session User A might create a new order and be allocated order number 1234. User B comes along at about the same time, creates an order, and is allocated order number 1235. Since the two actions originally happened very close together timewise, they are going to happen close together during the replay. However, since the environment that we are testing is a timesharing environment, it is possible that User B's robot will issue the call for an order number before that of User A, and the order numbers would be allocated 1234 to B and 1235 to A.



This may cause unexpected problems with subsequent statements. The original client may store the order number locally and then issue updates to order that it has no permission to change, etc. It is all dependent upon how the client is built. However, having said that, I have seen the robots built by this system used very effectively, despite getting the occasional error.

I was recently at TechWave and attended Rob Verschoor's interesting talk on Advanced DBA tricks. Rob also has a script for extracting the auditing data and putting it back together much like this one does. Rob's script is available at his download site (http://www.sypron.nl/audit.html) and does some different things to the reconstructed SQL for other types of analysis of the code. ■

Using PowerBuilder to Display Data for Web Applications

By Jay Hunt

A continuing series of articles on using the Datawindow to build applications for the Web

Jay Hunt is a Principal Architect with Automated Data Sciences. He can be reached at djayhunt@charter.net

Now that TechWave 2000 is in the history books, most of us are looking towards next year's conference event in San Diego or want to expand upon the knowledge we gained this year. Every year the number of sessions dedicated to Internet solutions continues to grow. For us old-time PowerBuilder advocates, this can be a little disheartening. However, I am here to share one constant through time—the PowerBuilder Datawindow.

In this article I will demonstrate how those familiar with the PowerBuilder Datawindow can leverage existing skills in the move towards Internet development. It is the intention to build upon my last article (*ISUG Technical Journal*, 3rd Quarter), in which I laid the foundation for using PowerBuilder Non-Visual Objects (NVO) with PowerDynamo and Jaguar for Internet development. If you are not familiar with the techniques reviewed in the previous article, you can find it posted online at www.isug.com.

When developers first learn PowerBuilder, they are introduced to the Datawindow and its power. Undoubtedly at some point in the development process, we are introduced to the Describe and the Modify methods of the Datawindow. By using these commands, we have dynamic access to attributes and properties of the Datawindow at run time. Of particular interest to the Internet developer is the Datawindow "Data.HTML" property. This property not only contains the data, but also pre-built HTML code necessary to show this data in a web page.

The simplest way to begin to leverage Datawindow skills is to call a method on a Jaguar component we build as a PowerBuilder NVO, get the "Data.HTML" property, and display it in a web page. Let's see how this is done.

Creating a Tabular Datawindow

Let's use the NVO created in my previous article, "Six Steps to Integrating Jaguar, PowerBuilder, and Power Dynamo," and a new Datawindow. Please note that this example makes use of the EAS demo database that ships with Sybase Enterprise Application Studio. This first example requires a simple tabular Datawindow of all customers in the Customer table.

Create a simple tabular Datawindow with no retrieval arguments and call it *d_cust_list*. Now we must create a new method in our existing Jaguar Component NVO (n_isugexample); we'll call it *of_customer_list*. This function will be a public function, it does not accept any arguments, and should return a string. Add the code displayed in Figure 1, which first sets up the transaction object and then establishes a connection to the database.

```
// Profile EAS Demo DB V3
SQLCA.DBMS = "ODBC"
SQLCA.Database = "EAS Demo DB V3"
SQLCA.AutoCommit = False
SQLCA.DBParm = "ConnectString='DSN=
   EAS Demo DB V3;UID=dba;PWD=sql'"

Connect;

Datastore        lds_datastore
String                  ls_html

lds_datastore = Create datastore

lds_datastore.dataobject = "d_cust_list"

lds_datastore.settransobject (SQLCA)
```

```
lds_datastore.retrieve()
ls_html = lds_datastore.describe("DataWindow.Data.HTML")

Return ls_html
```

Figure 1

Creating a Datastore

The next step is to create a Datastore for us to work with. Once the Datastore is set up and associated with the proper transaction object, the data is retrieved. Note the line of code:

```
ls_html = lds_datastore.describe("DataWindow.Data.HTML")
```

This is the exact piece of code that will produce the HTML syntax of our Datawindow. The string syntax will be stored in the variable ls_html for return to the HTML page for display. This NVO should be deployed to Jaguar via the Jaguar project we set up in my previous article. Remember to generate the stubs and skeletons as well.

Creating the Web Page Display

Next, the display portion must be created. This is done in PowerDynamo via Dynascript. Open PowerDynamo and create a new web page called "isugdw.htm". Add the code from Figure 2.

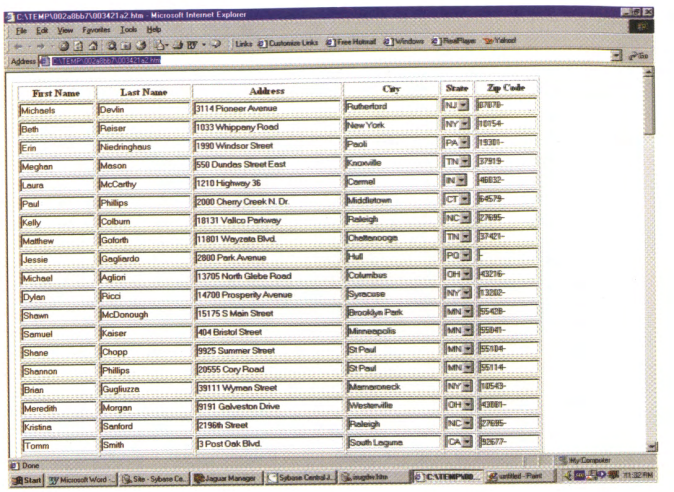
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">
<HTML>
<HEAD>
</HEAD>
<BODY>
<!--SCRIPT
  s_comp = "isugpackage/isugexample";
  s_jag = "iiop://localhost:9000";
  s_uid = "jagadmin";
  s_pw = "";

  s_jagcomp = java.CreateComponent (s_comp,s_jag,s_uid,s_pw);
  s_result = s_jagcomp.of_customer_list();

  document.writeln(s_result);
-->
</BODY>
</HTML>
```

Figure 2

When the page is browsed, the Datawindow is displayed as in Figure 3.



First Name	Last Name	Address	City	State	Zip Code
Michaels	Devlin	3114 Pioneer Avenue	Rutherford	NJ	07070-
Beth	Reiser	1033 Whippany Road	New York	NY	10154-
Erin	Niedringhaus	1990 Windsor Street	Paoli	PA	19301-
Meghan	Mason	550 Dundas Street East	Knoxville	TN	37919-
Laura	McCarthy	1210 Highway 36	Carmel	IN	46032-
Paul	Phillips	2000 Cherry Creek N. Dr.	Middletown	CT	06457-
Kelly	Colburn	18131 Vallco Parkway	Raleigh	NC	27695-
Matthew	Goforth	11801 Weyzata Blvd.	Chattanooga	TN	37421-
Jessie	Gagliardo	2800 Park Avenue	Hull	PQ	J
Michael	Aglion	13705 North Glebe Road	Columbus	OH	43216-
Dylan	Ricci	14700 Prosperity Avenue	Syracuse	NY	13202-
Shawn	McDonough	15175 S Main Street	Brooklyn Park	MN	55428-
Samuel	Kaiser	404 Bristol Street	Minneapolis	MN	55041-
Shane	Chopp	9925 Summer Street	St Paul	MN	55104-
Shannon	Phillips	29555 Cory Road	St Paul	MN	55114-
Brian	Guglluzza	39111 Wyman Street	Mamaroneck	NY	10543-
Meredith	Morgan	9191 Galveston Drive	Westerville	OH	43081-
Kristina	Sanford	21996h Street	Raleigh	NC	27695-
Tomm	Smith	3 Post Oak Blvd.	South Laguna	CA	92677-

Figure 3

Refining the Display with Browser Capability

At this point, the Datawindow does not look very good or offer much functionality. We can improve it by taking advantage of another Modify and Describe property on the Datawindow, "Browser." This is the part of the HTMLGen Datawindow property, and generates browser-specific HTML for Microsoft Internet Explorer and Netscape. This adds extra code to the generated syntax that takes advantage of absolute positioning available in these two browsers.

Create a new method in our NVO called "of_setbrowser." We will also have to create an instance variable "of type string" to hold the value for this browser, which we will pass in from the HTML side. This information is displayed in Figure 4.

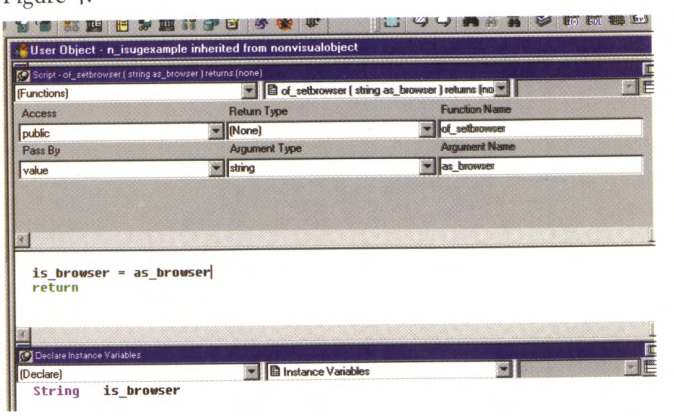


Figure 4

By adding an instance variable, we make the assumption that our Jaguar object requires variable persistence. This variable must be able to retain its value from one method call to the next; it is called a statefull object in Jaguar and requires us to

modify a deployment property in our Jaguar project. This is illustrated in Figure 5. Once this change is made, deploy the object to Jaguar. Again, remember to generate the stubs and skeletons.

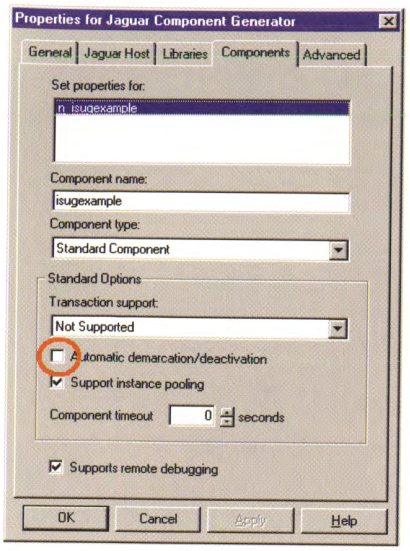


Figure 5

Before we test this, a word of advice: Start the personal web server from within Sybase Central. The code we are about to add to PowerDynamo will not work if the personal web server is not running. In addition we will not be able to simply browse the output. We must navigate to it via a PowerDynamo mapping such as http://localhost/ISUG/isugdw.htm.

Testing the Final Display

From within PowerDynamo, open our web page and add the code from Figure 6.

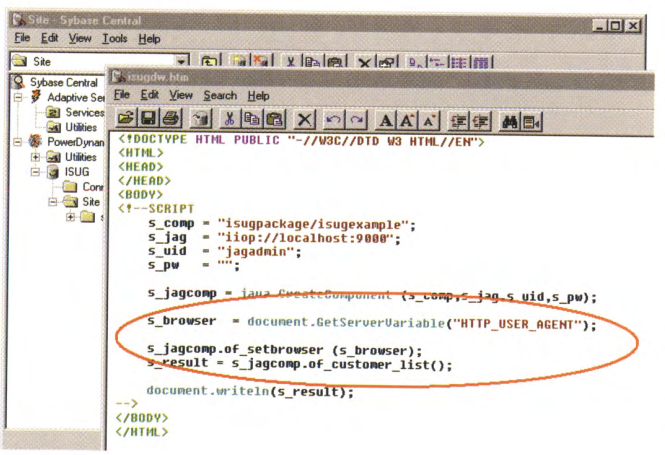


Figure 6

Make sure to place the call to the *of_setbrowser* prior to the *of_customer_list( )* method call. The of_customer_list method is responsible for providing us the HTML to display, and if we set the browser after this call, the code will not be created. Save the changes, open the browser and navigate to the HTML page. The output is displayed in Figure 7. Notice that the HTML looks a lot smoother with variable column width. Compare it to that displayed in Figure 3.

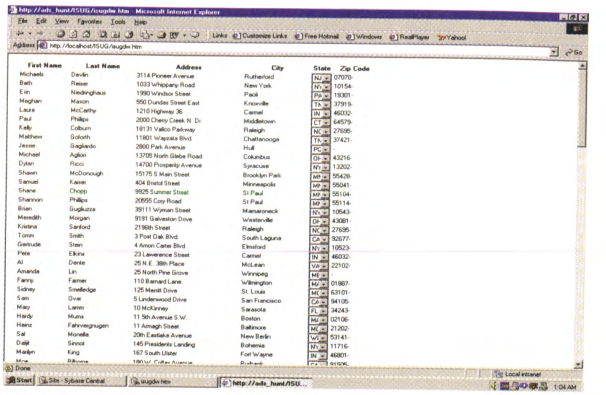


Figure 7

A Note on Creating a Global Object

Although we have the foundation to continue with our development efforts and the HTML Datawindow, consider this. The level of effort to code Datawindows under the methods illustrated in this example can be somewhat high. We might, however, have a repeatable process from which we could create a global object for HTML Datawindow display. Sybase has anticipated this need and provided us with such an object. This object is shown in Figure 8.

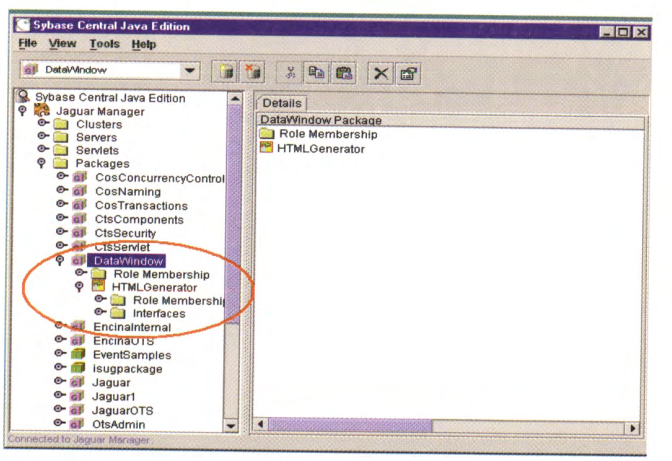
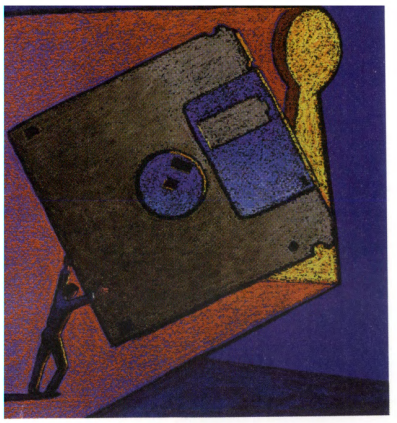


Figure 8

Coding for this object is not addressed within this article, but I do recommend that you review the source code in PBDWRMT.PBL, located in the Code Examples Subdirectory.

Conclusion

This article has demonstrated the basic principles and techniques necessary to make use of the HTML Datawindow from with Jaguar, PowerDynamo, and PowerBuilder. I recommend a further review of the code to learn about additional Describe and Modify properties that can enhance the use of the HTML Datawindow in your application. ■



Remote Access and Better ASE Security Using CGI Scripts

By Michael Peppler

Enhancing system security through CGI tools for the Unix server

In the last issue of the *ISUG Technical Journal* (3rd Quarter), Ed Barlow presented some interesting ideas on how to access ASE error logs and do server monitoring remotely using CGI scripts. His article focused on the techniques and perl code that can be used to achieve these results, but only briefly touched on the security issues raised by using CGI scripts, and by remote access from outside the company firewall. This article covers some of these issues, outlining tools that I use every day for remote access. I'm unfortunately very unfamiliar with Microsoft Windows, so I'll only cover Unix-related issues on the server side.

More and more work is being done via remote connections, using the Internet (or a corporate intranet) to access servers that are located in various places around the world. The ability to use the Internet for this allows us to work or provide support without needing to come into the office. For DBAs or system administrators who are on call, this is a tremendous improvement.

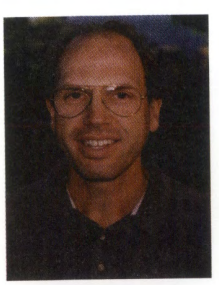
The use of CGI scripts to monitor servers is an example of tasks that can be automated to make the life of the DBA a little simpler. We'll cover the security issues of CGI scripts and remote access from inside out (i.e., from the CGI script itself to remote access techniques over the Internet). Note that my examples are written in perl, but the problems that are outlined are not language-specific.

Perl (and Other) CGI Script Security

The first thing you have to consider with CGI scripts is the impact of any security hole. Reducing the risk of such potential problems means running the server with a user ID that has as few privileges as possible. The default setup for most servers is to switch to the user *nobody* when the server is started by *root* (or "Super User"). However, if you start the server as another, non-root user, then you can't switch the user ID to *nobody*. Therefore, if you don't have root access to the machine and want to run a web server yourself, you should be extra careful, as all the CGIs will be running with your user ID.

One of the first CGI scripts that I wrote was a sysprocesses lister, which reloaded automatically every five minutes:

```
#!/usr/local/bin/perl -w
use strict;
use Sybase::CTlib;
use CGI;
my $dbh = new Sybase::CTlib 'sa', 'SomePassword', 'SYBASE';
my $q = new CGI;
print $q->header(-Refresh => '300; URL=' . $q->self_url);
print $q->start_html;
print "<table>\n";
my $restype;
$dbh->ct_execute('select * from sysprocesses');
```



Michael Peppler is a senior consultant with Data Migrations, Inc. He has worked with Sybase products since 1989, and wrote the sybperl prototype in 1990. He can be reached at mpeppler@peppler.org.

```
while($dbh->ct_results($restype) == CS_SUCCEED) {
        next unless $dbh->ct_fetchable($restype);
        my @cols = $dbh->ct_col_names;
        print "<tr>\n";       # print out the column headers
        foreach (@cols) {
                print "<td>$_</td>\n";
        }
        print "</tr>\n";
        my @data;
        while(@data = $dbh->ct_fetch) {
                print "<tr>\n";
                foreach (@data) {
                        $_ = 'NULL' unless $_;     # print explicit NULL
                        print "<td>$_</td>\n";
                }
                print "</tr>\n";
        }
}
print "</table>\n";
print $q->end_html;
```

This is a very simple script, and there isn't really very much that could go wrong with it: We don't accept any input from the user, and everything is hard-coded. However, for more complex scripts that accept user input via a form (or with parameters in the URL), there are issues that you should be aware of.

Problem #1: Vulnerable Form Variables

The most common problem is to take form variables (for example, an e-mail address) and pass this variable to a program that you run via a *system()* call:

```
$email = param('email');
system("mailx -s 'some subject' $email <somefile");
```

Looks innocuous, right? But what if the user entered:

```
michael@peppler.org; rm -rf *
```

in the form? As the $email variable is passed to a shell, the ";" character in the variable is interpreted as a command separator, with the result that **rm -rf *** gets executed in the current directory, with the credentials of the userid running the web server. Oops!

So how do you avoid this sort of thing? You must validate any input that you are going to use as input to other programs. In this case, you must make sure that all shell meta-characters (those characters that have special meaning to the shell, such as "*" and "?") are removed from the data before using it. The full list of meta-characters is

```
&;`'\"|*?~^()[]{}$\n\r
```

with the last two being line feed and carriage return.

With perl you can use the "taint" flag (a "-T" on the "#!" line invoking the script) to force you to "untaint" any data that is retrieved from some outside source by the program. Untainting is done by applying a regular expression match on the variable, and extracting only the part that is valid. So for the "$email" variable above, we could do something like this:

```
$email = /(\w+\@[a-zA-Z._-]+)/; $email = $1;
```

Problem #2: Variables Passed into SQL

The same sort of problem can occur if you allow CGI variables to be passed into SQL requests. For example:

```
my $name = $query->param('name');
$dbh->ct_execute("select ... from my_table where name =
  '$name'");
```

Again, it looks innocent enough. But what if the "$name" variable was:

```
michael' delete * from my_table where name != 'michael
```

Note the single quotes! Once the "$name" variable has been interpolated, the **ct_execute()** call will send a dual request, first the **select**, and then the **delete**, because of the way the quotes are placed.

Again, you need to guard against this by carefully handling quotes for any input parameters that are going to be used in a SQL request, especially if your CGI scripts use a privileged user ID for connecting to Sybase! (Hint: Create a login with minimal privileges to run all CGI queries, as that will limit your exposure to this sort of problem.)

If you are going to build more than a simple system, I strongly recommend using a user ID that has execute rights on stored procedures, and possibly select rights, but that is definitely not allowed to execute any **insert/update** or **delete** statements directly.

Using data input to subvert a CGI program to do something it was not designed to do is the most common type of security problem. There are others, of course. See the final paragraph of this article for some other sources of information on the subject.

Problem #3: User Authentication

There are probably some scripts that you don't want just anyone to run. You can use a couple of methods to check the credentials of the user running the script.

The simplest method is to restrict access to certain parts of your web server to specific client machines (using the IP address or the host name to determine this). For servers based on the NCSA source (Apache, for example) you would use something like this:

```
<Directory /full/path/to/directory>
 <Limit GET POST>
  order mutual-failure
  deny from all
  allow from 192.168.2 .peppler.org
 </Limit>
</Directory>
```

which would be located in the httpd.conf file and allow access from machines in the 192.168.2 address space, or in the .peppler.org domain. It's not very fine-grained, but it might be enough for what you need.

The next step up is to use server-level authentication. You use *htpasswd* to create a username/password combination, placed in the directory requiring authentication, and configure your web server to require it (possibly for certain directories only). When the user tries to access a URL in a protected directory, the browser pops up a username/password dialog box, and, when the user has entered a correct combination, the server allows access to the URL. This technique has the downside that the sessions stay valid for as long as the user's browser runs, but it is simple to set up.

A somewhat more complex scheme uses browser cookies to handle authentication. At the top of each script, you check for the existence of the cookie. If the cookie does not exist, you redirect the user to a login script, and then issue the cookie upon a correct login. To make this scheme reasonably secure, you should place something in the cookie that the user can't create on his own (usually this means using something like an MD5 hash, with some of the items passed to the hashing function kept secret from the user).

Lincoln Stein has written a pretty good module that does this. The module is described in Lincoln and Doug MacEachern's book, *Writing Apache Modules with Perl and C* (published by O'Reilly). I've used variations of this module for several large-scale public projects, and it works well.

Here's a simplified example of a cookie validation module:

```
package Cookie;
use strict;
use CGI::Cookie;
use Digest::MD5;

# gen_cookie() - takes a $user variable (could be a name or
# a user ID) and generates a cookie with an MD5 signature
# This cookie can then be used to authenticate requests
# via the val_cookie() subroutine.
sub gen_cookie {
        my $query = shift;    # $query is CGI.pm object.
        my $user = shift;

        my $secret = fetch_secret();
        my $hash = Digest::MD5->hexhash(join(':', $secret, time(),
                                        $user));

        return $query->cookie(-name => 'Ticket',
                -path => '/', -expires => '+1d',-value => {
                        user => $user, 'time' => time(),
                        hash => $hash
                });
}

# val_cookie() - checks to see if the "Ticket" cookie exists,
# and if it does checks to see if the MD5 signature is correct.
# Returns true/false, and the $user value or an error message.
sub val_cookie {
        my $query = shift;

        my (%cookie) = CGI::Cookie->parse($ENV{HTTP_COOKIE});

        return (0, "User has no cookie") unless %cookie;
        return (0, "User has no ticket") unless $cookie{Ticket};
        my (%ticket) = $cookies{Ticket}->value;
        return (0, "Malformed ticket") unless
                $ticket{hash} && $ticket{time} && ticket{user};

        my $secret = fetch_secret();
        my $newhash = Digest::MD5->hexhash(join(':', $secret,
                                        @ticket{qw/time user/});

        if($newhash eq $ticket{hash}) {
                return (1, $ticket{user});
        }

        return (0, "Invalid ticket");
}

sub fetch_secret {
        "some random string";   # should probably get from a file!
}

1;
__END__
```

Now, in each of your scripts you place a simple:

```
my ($val, $user) = Cookie::val_cookie($query);
if(!$val) {
    print $query->redirect("http://myhost/cgi-bin/login.cgi");
}
```

before actually doing anything, and login.cgi gets the username and password and validates this somehow (maybe a check in the database, or a check against a simple file kept locally) and sets the Ticket cookie:

```
my $cookie = Cookie::gen_cookie($query, $user);
print $query->redirect(-url => "http://myhost/main.html",
                       -cookie => $cookie);
```

This technique can of course be expanded or generalized. I tend to place the validation code in a module that itself initializes the *$query* (i.e., CGI.pm) object, as well as doing the authentication. The Digest::MD5 module, along with a very large list of other perl add-on modules, can be found on CPAN (see below).

You should be aware, however, that any authentication scheme you use will pass passwords (and user names) in clear-text between the browser and the web server (unless you use a secure server with SSL, of course). This may be a problem if the data has to travel over the Internet. Luckily, there are solutions to that problem too.

Problem #4: Remote Access to ASE and Other Servers

So you'd dearly like to be able to access the corporate ASE server(s) from remote locations. You could always ask your network operations folks to open port 4100 (or whatever port your Sybase server listens to) on the firewall, maybe limiting the source IP addresses that are allowed access to that port. My guess is that the network ops people are not going to like that solution very much. There is a much better solution, which only requires opening one port (22) and subsequently allows access to any internal service via an encrypted channel.

This solution uses the SSH (secure shell) protocol. SSH encrypts all communication using RSA (or various other methods) and allows you to do port forwarding. This means that you can tell SSH to forward port 4100 on the ASE server host to port 4100 on your local machine. You then create an interfaces file on your local machine that references port 4100 on host "localhost" and voila! You can access the corporate ASE server as if it were on your local machine, and all the traffic is encrypted between your machine and the host that you SSH to.

Port forwarding can be used for other things as well; for example, to give you access to your POP3 mailbox, or access to a web server that is not visible outside the corporate network.

SSH exists in a couple of implementations and there are clients for Windows, MacOS, and of course all the flavors of Unix. Port forwarding syntax for the Unix command-line client goes like this:

```
ssh -L <local port>:<remote host>:<remote port> <host>
```

So, say ASE runs on port 4100 on corpdb.peppler.org. I can do the following:

```
ssh -L 4002:corp.peppler.org:4100 ssh.peppler.org
```

This will connect to ssh.peppler.org, and create a connection between port 4002 on the localhost to port 4100 on corpdb.peppler.org. I now create an interfaces file with the appropriate entry using sybinit or dsedit, making sure to use "localhost" for the hostname, and I can now use all the normal clients from the local machine.

For Windows or Mac SSH clients you need to check the "Advanced" tab (or option), which will open a dialog box where you can specify the forwarding information.

Other Security Resources

These are some of the most glaring security problems that you should guard against. The list is by no means exhaustive, so I recommend reading Stein's Security FAQ and any other security documents you can lay your hands on! The list includes:

- Lincoln Stein (author of the CGI.pm module) maintains a CGI security FAQ at http://www.w3.org/Security/Faq/.
- CPAN - Comprehensive Perl Archive Network at http://www.cpan.org/. The archive is searchable.
- SSH: Commercial versions at http://www.ssh.com. OpenSource versions at http://www.openssh.org. ■

Time Translation and Quality Coding Methods in the Sybase Database

By Mark Gearhart

The first of a two-part series on translating time zones and displaying temporal data across Daylight Savings Time intervals

Historical energy accounting applications for electric utilities require the ability to manage data which is collected over time. In our applications, we use Sybase ASE to sample data for energy sales and purchases, transmission line voltages, circuit breaker statuses, power plant outputs and power system apparatus. This data is sampled on a periodic basis and stored as timestamped, numeric values.

In the beginning, designing and implementing a historic database to store this data was just another modeling exercise. We planned to get the requirements from the users, design the normalized database, implement it, and put the operational system into production. However, after noting the impact of Daylight Savings Time (DST) in disrupting an otherwise continuous sequence of hours throughout the year, we realized that the job was not just a simple matter of sampling and storing data. A single power plant which exports energy can easily generate in excess of $100,000 in sales every hour, so it was very important to track both the repeated 2 a.m. hour in the fall as well as the skipped 2 a.m. hour in the spring to avoid inaccuracies or even lost revenue due to the inability of the database to store repeated hours.

A further complicating factor introduced the possibility for confusion over time zones. Ultimately, we would need to store and support access to data originating from multiple time zones. Energy bills are often discounted by as much as 30% during the off-peak hours of 11 p.m. to 7 a.m., so it was very important to correctly handle queries issued from users in different time zones so as not to confuse relative time differences between geographic locations.

To complicate matters even further, we were also proposing to add an attribute, called *quality*, to each piece of data as it traveled from a meter located at a power plant, into the system, and ultimately into aggregate calculations for daily, monthly and yearly *max*, *min*, *sum*, and *averages*. This quality would be used to note unusual characteristics of the data, such as whether it was incorrectly sampled, whether it was altered after receipt, and so on.

Having two distinct design issues to deal with—time and quality—presented an opportunity to develop the methods presented in this paper for both temporal and bit-wise database operations using Sybase. Taken together, our solution provided the mechanisms needed to present data for any time zone, in a useful time and date format, and with a quality which describes the validity of the data. We successfully implemented these solutions on several multi-gigabyte, geographically dispersed energy accounting systems.

The first half of this article describes how we went about establishing the parameters of the problem and setting up global conversion to local time zones.

Mark Gearhart is an independent systems engineer specializing in product development for energy management systems. He can be reached at mark@mgearhart.com.

Problem Areas and Proposed Solutions

To begin with, the most fundamental requirement for the application was to provide a web page (Figure 1) which displays the megawatt-hour output of electric power from four generating units at a power plant:

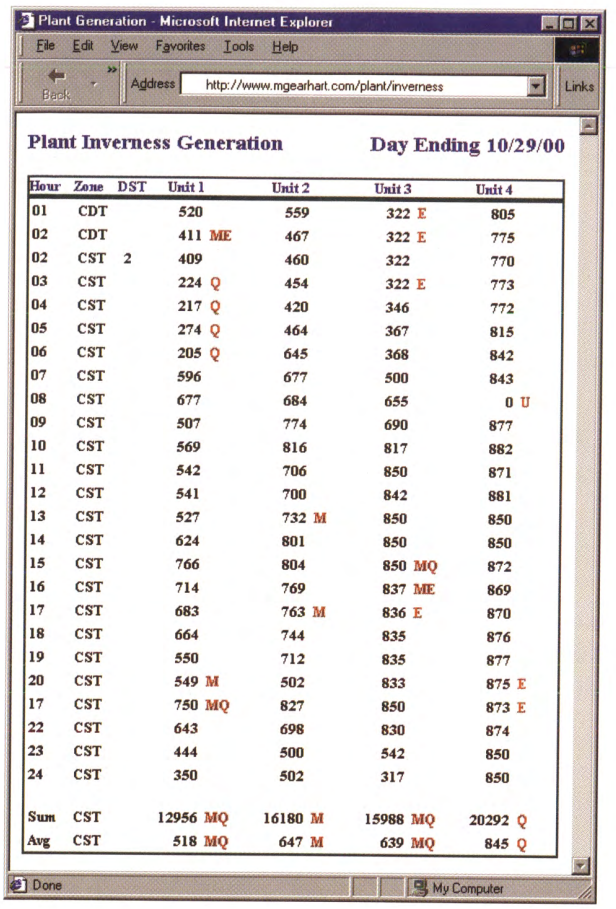


Figure 1. Initial web page for plant generation

Historical accounting systems require the ability to manage data which is collected over time. In the electric utility industry, data for energy sales and purchases, transmission line voltages, circuit breaker statuses, power plant outputs, and power system apparatus are sampled on a periodic or event-triggered basis and stored as timestamped, quality-coded numeric values. In our database model, the relation **HourData** is a set of records (tuples) that store a continuous history for each object. Each such record has a time-invariant key called **VarNum** and three time-variant attributes called **Time**, **Value**, and **Quality**. When we say *continuous*, we are describing a model in which each row is adjacent to each other with respect to time, so that the data for each object is stored as an hourly sequence.

If each generating unit at each power plant is identified by the **VarNum** attribute, then the query to select hourly data for variable 2953, in this case "*Plant Inverness, Unit 1*", could be written as follows:

```
select Time, Value, Quality
from HourData
where Time between "6/29/00 00:00" and "6/30/00 00:00"
and VarNum = 2953
order by Time
```

Figure 2. Initial query for Plant Inverness, Unit 1

Although this query is very straightforward, we find that it is insufficient for our application. For example, if the query is executed on the Daylight Savings Time day, which row is the first 2 a.m. and which is the second? Suppose we want to use 2 a.m. as a search argument? Which 2 a.m. do we use?

Since the 2 a.m. timestamps are identical, an **order by** clause will not determine the correct order. We could store the second time as 1 millisecond past 2 a.m., but this is clearly inaccurate and complicates **group by** clauses. We could introduce a Daylight Savings Time attribute at the expense of disk space, but this also introduces an extra composite in the primary key which is not desirable. Moreover, these solutions put a burden on applications which insert or update data to know exactly when the Daylight Savings Time intervals occur.

Another problem is Sybase's timestamp used for midnight. Our users want to see a web page that considers the last hour of the day as the 24th hour of the current day, whereas Sybase's implementation of the **datetime** data type returns the last hour as the 0th hour of the next day. Given this, how do we translate an internal Sybase timestamp into a useful time for query and display purposes?

Still another problem deals with the time zone used when storing the data versus the time zone of the user accessing the data. If a value is stored relative to midnight USA Eastern time, does a user in the USA Central time zone access the data as a midnight value or an 11 p.m. value? Suppose the value is replicated? If the replicate site resides in another time zone, what time do we store for the copy?

Beyond this, we are faced with an altogether separate issue, which nevertheless must be solved for the entire application to be useful. We must consider that each value has a quality associated with it. In fact, the quality actually represents many simultaneous characteristics in effect for the value, such as whether it has a range error; whether it has been manually entered via the web page after being sampled; whether it

was estimated from a previous hour sample, and so on. Taken to the final implementation, the quality of each value also affects its inclusion in daily, monthly, and yearly aggregates. For example, we have a rule which states that the daily sum must exclude any inputs whose quality indicates that the reading is bad. This implies the use of special conditions against the Sybase functions **max()**, **min()**, **sum()**, and **avg()** for omitting certain inputs as well as for propagating many input qualities into a single output quality (i.e., an aggregate bit-wise "or" operation, which Sybase does not support).

Extending the data model to support time-varying data with quality codes might imply that the implementation of the model—in this case the Sybase ASE product itself—must be changed. Since this is not feasible insofar as the average application developer is concerned, a corresponding change must lie elsewhere. One approach to achieve some support for handling time-varying, quality-coded data might be to build temporal and quality functionality into the database application, using the extensibility inherent to the Sybase Open Server. This could result in substantial man-hours needed to implement the temporal and bit-wise framework in code.

Another approach might be to implement abstract data types for time and quality. This assumes, of course, that the DBMS supports such a facility. This is not possible using a pure relational platform provided by Sybase.

The last approach, which we have implemented, is to use the type **datetime** supplied in a different temporal context and build the support for time translation into our applications. We also use the type **integer** in the context of a bit mask and likewise build our own support for quality codes. For this approach, our only assumption is the data type **datetime** and **integer**. As shown in the following sections, we have built the temporal and quality operations into the application itself through the use of *translation relations*, *views*, and *stored procedures*.

UTC Time and Daylight Savings Time

Times are stored in UTC (Universal Time Coordinated) time rather than local time. UTC time presents an alternative in which timestamps are not only unique, but can be distinguished during Daylight Savings Time intervals. UTC time (also called GMT or Greenwich Mean Time) is a zero time zone which extends 7.5 degrees east and 7.5 degrees west of the Greenwich meridian, 0 degrees longitude. Local standard time can be determined for any UTC time for anywhere in the world by adding 1 hour for each time zone counted in an easterly direction from one's own time zone, or by subtracting 1 hour for each time zone counted in a westerly direction.

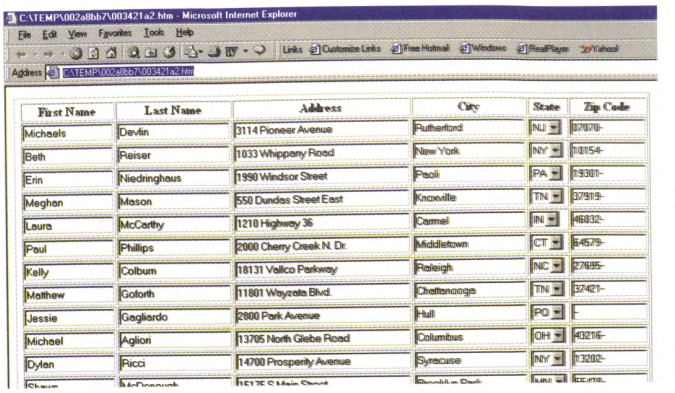


Figure 3. Standard Time Zones of the World

However, not all countries offset their local time by a multiple of one hour. For example, the central time zone of Australia is offset from UTC time by 8 hours, 30 minutes. Countries that adhere to the international time zone system adopt the zone applicable to their location. Some countries establish time zones based on political boundaries or adopt the time zone of a neighboring unit. A country like China, for example, spans five time meridians and yet references just one time zone. Moreover, electric utilities may sometimes choose not to follow the time zones established by either the meridians or political boundaries. For example, some electric utilities view all data as if it were in one time zone, even though the power grid may span several meridians or political boundaries.

For all or part of the year, some countries advance their time by one hour, thereby using more daylight hours each day. For this adjustment, an hour is added in the spring, thereby skipping an hour and producing a 23-hour day. Then, in the fall, an hour is subtracted, which repeats an hour and produces a 25-hour day. There are also cases throughout the world in which Daylight Savings Times may adjust by more than one hour. In Israel, for example, an adjustment of two hours is performed in the spring and fall.

Data is stored in UTC time to ensure uniqueness. In UTC time, there are always 24 hours in a day. For any location in the world the local time can be calculated by adding or subtracting a certain number of minutes, which may vary from one day to the next depending on the Daylight Savings Time adjustment. We recognize that one would like to access and present data oriented to local time rather than Greenwich England. Accordingly, we have implemented the methods needed to convert between these time references.

Translating UTC to Local Time

We revise our initial model of the **HourData** relation in order to support **UTC time**. In this revision, the **Time** attribute is replaced by a **UTCTime** attribute. When selecting data from **HourData**, the additional relations **TimeZone**, **UserTimeZone**, and **TimeTran** (Figure 4) are used to determine the time adjustment to be applied to every timestamped value in order to convert its UTC time to a local time.

Zone	ZoneDesc
1	USA Eastern
2	USA Central
3	USA Mountain
4	USA Pacific
30	China

TimeZone Relation

ServerLogin	Zone
mrgearha	2
mstorres	2
nlgaus	1
alchoy	30

UserTimeZone Relation

Zone	UTCStart	UTCStop	DST	Offset	ZoneAbbr
1	10/31/99 08:00:00.000	04/02/00 06:59:59.999	""	-300	EST
1	04/02/00 07:00:00.000	10/29/00 06:59:59.999	""	-240	EDT
1	10/29/00 07:00:00.000	10/29/00 07:59:59.999	"2"	-300	EST
1	10/29/00 08:00:00.000	04/01/01 06:59:59.999	""	-300	EST
2	10/31/99 09:00:00.000	04/02/00 07:59:59.999	""	-360	CST
2	04/02/00 08:00:00.000	10/29/00 07:59:59.999	""	-300	CDT
2	10/29/00 08:00:00.000	10/29/00 08:59:59.999	"2"	-360	CST
2	10/29/00 09:00:00.000	04/01/01 07:59:59.999	""	-360	CST
30	NULL	NULL	NULL	+480	NULL

TimeTran Relation

Figure 4. Relations used to convert between UTC and local time

The **TimeZone** relation contains an entry for every time zone used by the application, and potentially every time zone in the world. Our particular application uses the USA Central Time zone, and we have listed other time zones in Figure 4 as examples only. The **UserTimeZone** relation contains an entry for every Sybase user, specifying the time zone in which they reside.

The **TimeTran** relation describes the behavior of time. There exist many application domains which need access to not only the most recent snapshot in time, but also to past and even future events, so the notion of time translation must be extended to cover all temporal occurrences for which usage is relevant. Examples of such applications are financial applications (e.g., the history of stock market data), insurance applications (e.g., when policies are in effect), reservation systems (e.g. ,when hotel rooms are booked), medical information systems (e.g., patient records), and decision support systems (e.g., planning for future contingencies). **TimeTran** contains a description of time which can encompass these domains. This description is organized by time zone. It can be seen from the example in Figure 4 that the USA Central time zone contains a row for each interval whose offset from UTC time is different from the next interval, for a period spanning 10/31/99 09:00 to 04/01/01 07:59:59.999. Time zones can range from something as simple as China, whose offset from UTC never changes, to a Daylight Savings Time zone such as USA Central, which must be configured for many specific intervals.

UTC-to-local time translations are performed according to the interval and offset configured for the time in question. For *mrgearha*, who resides in the USA Central time zone, time translation for the spring and fall Daylight Savings Time days would proceed as follows:

For this UTC Time:	add this offset:	to produce this local time:	and DST indicator:
04/02/00 06:00:00	-360	04/02/00 00:00:00	""
04/02/00 07:00:00	-360	04/02/00 01:00:00	""
04/02/00 08:00:00	-300	04/02/00 03:00:00	""
04/02/00 09:00:00	-300	04/02/00 04:00:00	""
10/29/00 06:00:00	-300	10/29/00 01:00:00	""
10/29/00 07:00:00	-300	10/29/00 02:00:00	""
10/29/00 08:00:00	-360	10/29/00 02:00:00	"2"
10/29/00 09:00:00	-360	10/29/00 03:00:00	""

Figure 5. Applying UTC time offsets to produce local times

When data is stored into the database, the time zone of the originating data source is chosen. This can present some confusion for users who access this data from different time zones. If the data is stored in one time zone (Central Time, for example) and is accessed by a user in another (Pacific Time), the offset configured for the user is applied upon selection. Therefore, while the user in Central Time (same as the storage time zone) sees a value with a midnight timestamp, the user in Pacific Time would see that same value with a timestamp of 10 p.m., not midnight. Both of these times are potentially correct, depending on the application. Initially, however, we have applied our implementation to a collection of sites whose data and users are in a single time zone. In the

next issue, we propose two possible solutions for handling geographically dispersed data selection.

Once the offsets have been configured, data may be selected from **HourData** by converting the **UTCTime** attribute into local time. This conversion of time is coded as part of the query. In addition, the local hour-ending time is included in the result. For data with other frequencies, derived attributes for day-ending, month-ending, and year-ending can also be included. The query to select a local time is composed as follows:

```
select
 UTCTime   = UTCTime,
 LocTime   = dateadd(mi,Offset,UTCTime),
 LocTimeEnd = convert(char(9),dateadd(ms,-2,dateadd
 (mi,Offset,UTCTime)),1) +
        right("0"+convert(varchar(2),datepart(hh,dateadd(
        ms,-2,dateadd(mi,Offset,UTCTime)))+1),2) + "00",
 HourEnd   = right("0"+convert(varchar(2),datepart(hh,dateadd(
        ms,-2,dateadd(mi,Offset,UTCTime)))+1),2),
 DST       = TimeTran.DST,
 ZoneAbbr  = TimeTran.ZoneAbbr,
 VarNum    = VarNum,
 Value     = Value
from HourData,TimeTran,UserTimeZone
where UTCTime >= isnull(UTCStart,UTCTime)
and  UTCTime <= isnull(UTCStop,UTCTime)
and  TimeTran.Zone = UserTimeZone.Zone
and  ServerLogin = suser_name()
and  UTCTime > "10/29/00 05:00"
amd  UTCTime <= "10/30/00 06:00"
and  VarNum = 2953
order by UTCTime
```

Figure 6. Query used to select a local time

To calculate an hour-ending time, 2 milliseconds (the smallest unit which will affect a Sybase **datetime** data type) are subtracted from the local time to bring it into the previous hour. This produces an hour from 0 to 23. Then, one hour is added to the hour part in order to produce an hour-ending quantity. Since there are time zones which do not have Daylight Savings Time intervals, the **UTCStart** and **UTCStop** attributes in **TimeTran** may be null. The China time zone is an example of this. Therefore, an *isnull()* function is used when joining to **TimeTran**. The desired time zone is chosen based on the Sybase user, and is also coded as part of the query by using the *suser_name()* function. The local hour is returned as a **char(2)** field so that a leading 0 can be prefixed to the single-digit hours 1-9, thereby producing a displayable value from "01" to "24".

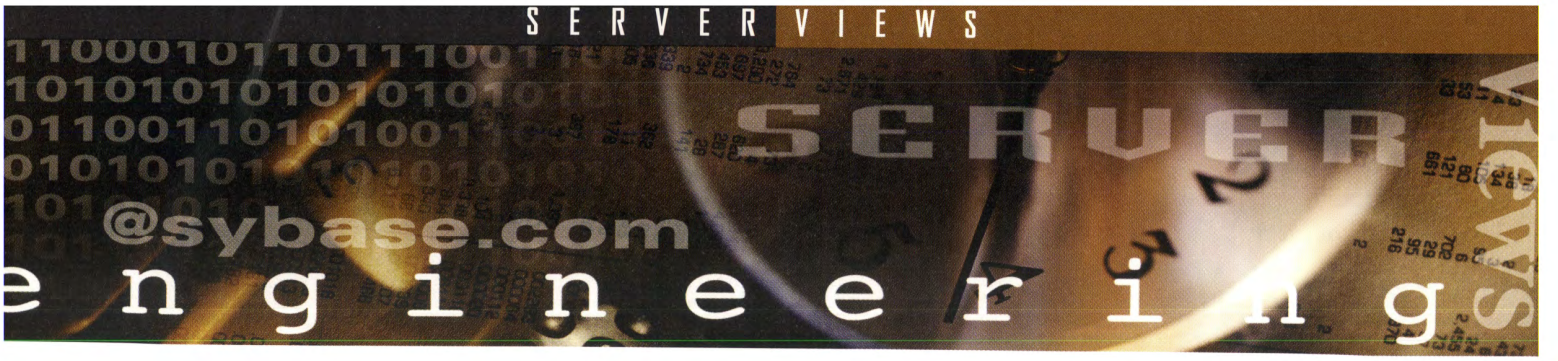
Having formulated the translation from UTC to local time within the query, results are returned as follows:

UTCTime	LocTime	LocTimeEnd	HourEnd	DST	ZoneAbbr	Value
10/29/00 06:00:00	10/29/00 01:00:00	10/29/00 0100	01		CDT	520
10/29/00 07:00:00	10/29/00 02:00:00	10/29/00 0200	02		CDT	411
10/29/00 08:00:00	10/29/00 02:00:00	10/29/00 0200	02	2	CST	409
10/29/00 09:00:00	10/29/00 03:00:00	10/29/00 0300	03		CST	224
10/29/00 10:00:00	10/29/00 04:00:00	10/29/00 0400	04		CST	217
10/29/00 11:00:00	10/29/00 05:00:00	10/29/00 0500	05		CST	274
10/29/00 12:00:00	10/29/00 06:00:00	10/29/00 0600	06		CST	205
10/29/00 13:00:00	10/29/00 07:00:00	10/29/00 0700	07		CST	596
10/29/00 14:00:00	10/29/00 08:00:00	10/29/00 0800	08		CST	677
10/29/00 15:00:00	10/29/00 09:00:00	10/29/00 0900	09		CST	507
10/29/00 16:00:00	10/29/00 10:00:00	10/29/00 1000	10		CST	569
10/29/00 17:00:00	10/29/00 11:00:00	10/29/00 1100	11		CST	542
10/29/00 18:00:00	10/29/00 12:00:00	10/29/00 1200	12		CST	541
10/29/00 19:00:00	10/29/00 13:00:00	10/29/00 1300	13		CST	527
10/29/00 20:00:00	10/29/00 14:00:00	10/29/00 1400	14		CST	624
10/29/00 21:00:00	10/29/00 15:00:00	10/29/00 1500	15		CST	766
10/29/00 22:00:00	10/29/00 16:00:00	10/29/00 1600	16		CST	714
10/29/00 23:00:00	10/29/00 17:00:00	10/29/00 1700	17		CST	683
10/30/00 00:00:00	10/29/00 18:00:00	10/29/00 1800	18		CST	664
10/30/00 01:00:00	10/29/00 19:00:00	10/29/00 1900	19		CST	550
10/30/00 02:00:00	10/29/00 20:00:00	10/29/00 2000	20		CST	549
10/30/00 03:00:00	10/29/00 21:00:00	10/29/00 2100	21		CST	750
10/30/00 04:00:00	10/29/00 22:00:00	10/29/00 2200	22		CST	643
10/30/00 05:00:00	10/29/00 23:00:00	10/29/00 2300	23		CST	444
10/30/00 06:00:00	10/30/00 00:00:00	10/29/00 2400	24		CST	350

Figure 7. Results from the local time query

As shown, the last hour of the day is returned as Hour 24 rather than Hour 0, and all other hours remain unaffected. Note from Figure 7 that we can now identify the first and second 2 a.m. hours by the "2" indicator, which denotes the *second* 2 a.m. hour. Thus, both the Daylight Savings Time and hour-ending problems are solved. In addition, the UTC timestamps in the base relation are unique and do not require either inaccuracies or an extra DST attribute.

The second half of this article will appear in the *ISUG Technical Journal*, 1st quarter issue, 2001, covering the methods for attaching quality codes to data, accessing data through views, selecting data into multiple columns, using quality codes to compute aggregates, and using UTC time for inserts and updates. ■



A Preview of ASE 12.5

By Ian Smart

Recently I achieved a great milestone in my life, as I have now been using ASE (or SQL Server as it was called then) for ten years. I started as a DB2 consultant for an systems integrator and we received an invitation to bid from a large UK customer. The company in question had already made the decision to use a Sybase SQL Server to build what was going to the second-largest Sybase database on the planet—a huge 26GB! Since I was free that day, my director gave me the document with the comment, "Hey, Ian, you know relational databases, have a look at this. It's Sybase, but they are all similar." Suffice it to say that after a five-month bid process we won the business, and the rest, as they say, is history. (My director's fate was not so glamorous; he is now a director at Oracle.)

But was my ex-director right? Well, I guess that we all subscribe to Ted Codd's 13 Rules and the premise that you use SQL to state what data you require rather than how to get it. In reality, as we know, there is a great deal more to it than that. To begin with, there are issues such as scalability and performance. This is an area where Sybase has always prided itself in being the leading player. The recent World Record breaking TPCC number of 156,873 tpmC at a price of $48.81 per tpmC is the latest in a long line of such records that keeps up this proud Sybase tradition. This is especially impressive when you consider that Oracle's best results of 115,395 tpmC were achieved at over twice the cost per tpmC.

So, is it enough to be the fastest? Well, no. I was recently with one of the CS&S managers in the UK and asked him about a particular customer. He replied that they didn't really get many cases from customers like this one these days. He looked wistfully out of the window and said that the days of CS&S spending their time answering simple cases were long gone. With the huge improvements in the quality of the code and the improved education of the user base, they got very few, but far more complex, cases these days.

I was also talking with a user recently who had just been round the cycle of revalidating their choice of database vendor and had decided that ASE was still the database for them. His reason was a surprising one. He said that one of the greatest sales tools that Sybase has is its educational materials. "Look at the backup and recovery modules," he said. "With Sybase you learn everything you need in one afternoon; with your competitor it took over three times that long, and they only covered part of the material. If it is that complicated to do such an essential task, think what everything else to do with that product is like."

So, is it enough to be the fastest, most scalable, most cost-efficient, high quality, and easiest to use? Well, no. By the time you read this article, ASE 12.5 will be well into beta testing. ASE 12.5 builds on the portal-enabling capabilities of ASE 12.0, such as support for Java as both a means of defining complex datatypes and user defined func-

tions, Distribution Transaction Management's support for XA and DTC, and High Availability support for use in active-active clusters. So what does ASE 12.5 provide? The following description, which was co-authored with ASE Product Manager Sumit Kundu, sets out to whet your appetite.

The Portal-Ready Database for Business on the Internet

Back in 1990, the SQL Server was focused on meeting the needs of users such as myself who wanted to perform OLTP transactions on tightly identified data in as rapid a fashion as possible. By the mid-'90s, the requirement for decision support meant that it was not sufficient to simply provide rapid access to small subsections of the data. Sybase added parallel queries and large I/O support. As the '90s drew to a close, IT moved from a propensity to build applications to buying packaged solutions. ASE accommodated this change in the new data storage and locking models. Finally, as we entered the new millennium, the requirement for portals provided a new set of challenges for ASE 12.0.

The next major focus across IT that we are now seeing is the growth of global e-business. Analysts see this providing 20% growth in the IT sector in the next year. Therefore, once again ASE needs to rise to the challenge. In ASE 12.5, Sybase is providing the database for e-business portals. However, since we are building on the investments we have made in the past, coupled with other new improvements we have made in existing areas of the server, this is not being done at the expense of the existing ASE users. Rather, these are enhancements that should enable all users to expand their ASE usage building upon their existing investment.

Productivity Enhancements

XML Query Services

To remain competitive, e-businesses need to ensure that the content that they publish on their portal is both up-to-date and tailored for the individual requirements of the user who is accessing it. In order to achieve this, web content must be stored via a means that can be enriched when it is retrieved, and which allows customization of the same data, depending upon who is accessing it.

One possible means of achieving these objectives is to store the content in the database as an XML document. By storing the XML as either a DOM object or breaking the data out into a table, the Java routines can vary the XML that is presented back to the client, depending upon that client's profile. ASE 12.5 extends the XML capabilities provided in ASE 12.0 by providing facilities to allow dynamic content generation using XQL (a "pseudo" standard). Using XQL, users will be able to specify queries and generate in-memory access structures that will enable the retrieval of the desired information.

These in-memory structures are optimized for "store once and query many times," but depending upon the user issuing the request, the data returned can be structured quite differently. Therefore, retrieval of selected XML-based information to a browser that can display the requested information can be both flexible and fast. Just one example of XQL:

```
create table XMLTEXT(xmlcol text)

-- Note: In XQL the elements are separated by '/'s
-- e.g. /bookstore/book/title

-- From all the documents in the table, get all the books
-- whose author's first name is Mary and return the xml
-- document

select
xml.Xql.query("/bookstore/book[author/first-name = 'Mary']",
   xmlcol)

from XMLTEXT
-- When the data isn't even in ASE this can still be used,
-- you can still find out who wrote that book on the web

URL xmlURL = new
URL("http://mybookstore.com/bookstore.xml");

String result = Xql.query("/bookstore/book/author",
xmlURL.openStream());
```

Support for Enterprise Java Beans (EJB)

In ASE 12.0, Java classes and methods could be used within ASE to provide user-defined datatypes and methods. In ASE 12.5, support for the Enterprise Java Beans model will enable rapid assembly of enterprise applications for pre-fabricated components that are typically transactional in nature. Due to the way that this has been implemented, developers are not forced to write such capabilities as session management, security, transaction, remove invocation and multi-threading, since these will be provided and can be switched on when required.

The EJB support will be cross-platform, allowing applications to be developed on one platform and be deployed elsewhere. Through the provision of EJB support within ASE, a number of benefits are derived—including improved performance for data-centric beans—by reducing overhead of network communication and enhanced security by limiting the bean to the secure RDBMS environment. In addition, the ability to utilize beans that were previously limited to the

middle tier of your architecture will enable a greater ROI on those components. The EJB support in ASE 12.5 will comprise the following features:

- Support EJB 1.1 specification, including stateless and stateful session beans and entity beans.
- The capability to deploy the beans using PowerJ. This will automatically generate server-side bean classes for the bean home interface. It will also support a textual deployment descriptor that specifies the transactional and security attributes of a bean among other attributes. The facility will automatically generate the Java class that extends the base deployment-descriptor class.
- The flexibility to turn off EJB support to reduce system resource requirements.

Support for SQLJ Standards

Sybase has been in the forefront of driving the standardization of Java/relational database technology. The outcome of that effort is SQLJ. This standard is comprised of three parts. Part 0 comprises embedded SQL in Java and is based on JDBC. Part 1 defines the standards for using Java static methods as SQL stored procedures and functions, and in Part 2, Java classes are used as SQL abstract data types.

ASE 12.5 will be fully compliant with Part 1 of the SQLJ standard. Apart from supporting Java stored procedures, other capabilities include support for output parameters and result sets. As you will recognize, partial support for Part 2 is already provided in ASE 12.0. Plans for the support of Part 0 are also well advanced. An example of SQLJ procedure:

```
create procedure isug_Journal
(in article_no int, out author char(40) )
dynamic result sets 1
language java parameter style java
external name 'Isug.author'
```

which executes the Java:

```
public class Isug {
        public static void author(int ano, String author)
   throws  SQLException
  {                 'JDBC code'
    if (ano==4) { author = 'Ian Smart'; }
                  else { author = 'Someone Else'; }
.... more code ....}
  }
```

Web Content Management

Much of the existing content that enterprises wish to share with their customers across the web is stored in non-relational form outside an RDBMS. Typically, such data is stored in operating system supported files. These are difficult to manage, especially when the content they are storing is increasing so rapidly. ASE 12.5 provides new capabilities to manage external data stored in operating system files. Utilizing ASE's CIS capabilities, a proxy table can be created to map to all the files in a specified directory. Using normal SQL commands against that proxy table, files can be created, deleted, read and written. In addition, permission attributes of a file can also be altered. ASE/CIS, in coordination with the operating system's file system, will provide access control so that a managed file cannot be deleted using the command shell.

JVM Performance Enhancements

One of the features of ASE 12.0 that really captured the imagination of the user community was the support for Java within ASE. In presenting this functionality to user groups around Europe and the U.S., it was very interesting to see how many different uses were suggested for this functionality—ideas ranged from ways of using the XML facilities that were enabled using this feature, through to the provision of varying report line formats for different reports all using the same data in the same table, but which specified the required report formatting as a parameter.

In ASE 12.5, this functionality is used to support the XML query services discussed earlier. With this Java focus, the performance of the embedded Java Virtual Machine (JVM), the engine to execute Java code, becomes crucial. In ASE 12.5, the embedded JVM will leverage Just-In-Time (JIT) technology to provide a performance boost. Changes in JVM technology will be completely transparent to end-users. The embedded machine will compile Java code to native machine code, as and when necessary to provide superior performance.

In addition to performance enhancements, ASE 12.5 will be J2EE compliant with support of the Java2 platform.

Large Database Pages and Unicode Support

Due to the globalizing effect of the web, it is no longer sufficient to view your e-Business solution as only needing to work in one time zone or language. Many software applications, particularly ERP applications, are being used worldwide and contrary to the views of many of my fellow countrymen, just working in English is not an option. There is a need to

offer support for a broad range of languages, which raises the requirement to standardize on uniform encoding of all languages. ASE 12.5 will support portal providers in fulfilling this important requirement through the provision of UCS-2 Unicode encoding of character data.

To facilitate UCS-2 encoding, ASE 12.5 will support 4KB, 8KB and 16KB page sizes in addition to the 2KB-page size. As a direct result of this, it will also be possible to use increased row, column and index key sizes. These larger pages will reside in larger extents, which will allow the use of 128K cache pools using 128K I/Os. This will be of particular benefit to users with large databases and/or queries that need to scan large amounts of data that cannot all fit in cache.

High Availability (HA) Enhancements

As mentioned above, continuous availability 24x7x365 is a real requirement in a truly global marketplace. Therefore, downtime, whether planned or unplanned, is simply unacceptable. EP 1.0 used the ASE 12.0 HA functionality as part of the means to make it the world's first truly non-stop portal. In addition, the ASE HA solution was implemented without the need for performance sapping methods of handling distributed locks, but instead simply utilized two fully functional, SMP servers in a dual-node cluster environment, communicating via CIS, to support active-active and active-passive failover.

With ASE 12.5, the reach of Sybase's continuous availability solutions will be further extended to support industry-leading cluster solutions from Veritas and proprietary business continuance solutions from EMC. In addition, ASE 12.5 will also support transportable databases, where a database can be unmounted from a server and mounted to another ASE. This allows a production database for OLTP to be moved to another server dedicated for DSS, without the need for the ASE server to be stopped and started, enabling access to all the other databases owned by both servers to continue supporting the portal users.

Internet Security Services

For prospective customers to use your e-business site, they need to have confidence that your portal is secure. However, true security also requires secure mechanisms to transport the data once it has entered your n-tier architecture. In ASE 12.5, the existing security capabilities will be enhanced for both communication and storage through support of the Secure Socket Layer (SSL) protocol and new capabilities providing user-level access control of data. The SSL support can also be used to provide Public Key Infrastructure (PKI)-based security that uses public-key cryptography, digital certificates, and Certificate Authority (CA) as part of an enterprise-wide network security architecture. The security infrastructure will be able to create public and private key pair for a user and issue a certificate-signing request that is fulfilled by the CA. Once the certificate has been issued and stored, a user should be able to connect to ASE without additional passwords. In addition, the SSL can also be used with digital certificates for authentication, encryption, and end-to-end security.

Row-Level Storage Security

As well as the new security mechanisms ensuring server access and the existing capabilities for granting and revoking permissions to tables and columns, ASE 12.5 will also support security mechanisms associated with the rows of data that are stored. Through the provision of Fine Grade Access Control (FGAC), SSOs will be able to specify that certain users can only retrieve specifically included, or not excluded, portions of a table depending upon who the user is. The solution is entirely server-based, making it unnecessary to write any filtering code at the application level to deal with security.

The solution involves the creation of an access rule by the user or application. This rule is bound to a data type, which, when used as a column type, applies the rule to all operations on the table containing the column. This storage level security is not limited to a specific SQL table, but can be applied to all tables using the data type. In addition, default rules can be described for server-wide enforcement.

Data Confidentiality and Integrity

The protection of critical business data is essential while it is in transit to and from the user. ASE will use Secured Socket Layer (SSL) protocol to provide security over the communication channel, which will complement the user authentication and storage security discussed above. This protocol will encrypt network data, provide integrity checking, and authenticate clients and servers. The SSL protocol performs a handshake between client and server using PKI infrastructure to authenticate the client. A successful handshake provides the client access to all applications in the enterprise without requiring any other password.

Internet Directory Services

LDAP (Lightweight Directory Access Protocol) is fast becoming the standard for enabling anyone to locate organizations, individuals, and other resources such as files and devices in a

network. This capability is not limited to either the Internet or a corporate intranet. The LDAP protocol provides a mechanism to handle directory information that is used enterprise-wide.

ASE's Internet Directory Services will include the services of an LDAP Server to provide both server information (as a replacement for interfaces file), and user information, including identification, authorization, user profile, or any other type of application information. The solution will involve the integration of a full-use LDAP server with ASE. This integration may use ASE as a data store for directory information or could use other types of stores. However, the choice of a store will be transparent to the user, except for configuration purposes.

In addition to the integration itself, new capabilities will allow publishing of information kept in ASE tables to LDAP directories. This will allow customers to efficiently handle certain classes of queries, such as those involving user authentication. In addition, this LDAP server will be available for use with JConnect in an HA environment, removing the need for the external LDAP Server that is used with ASE 12.0.

Developing Portal and Internet-based Solutions Using ASE 12.5

In addition to the pure ASE features, the use of ASE with other Sybase products allows for even greater capabilities and flexibility to be obtained. In particular:

- ASE, through features like remote procedures and CIS/EnterpriseConnect, will provide access to all of the data stores that customers might already have, and that customers might acquire in the future. To enable transactionality across these datastores, ASE 12.5 builds on the ASTM functionality added in ASE 12.0, and will also support 2PC across heterogeneous databases using EnterpriseConnect.
- The high availability enhancements and engine online/offline features from ASE 12.0, coupled with the OpenSwitch server, can be used to provide 24x7x365 availability. In fact, when considered alongside abstract query plans (which reduce the impact of ASE versions upgrades), and high performance online backup and recovery (which reduce downtime), it can be seen that the majority of users availability requirements can be met very easily and without any single point of failure. In ASE 12.5, new features like quality of service with dynamic reconfiguration, transportable databases and solutions like EMC's TimeFinder with Quiesce Database will ensure that server downtime is further reduced.
- In combination with Enterprise Application Server, ASE's EJB support will enable customers to deploy e-commerce applications in a much more rapid fashion with greatly reduced development time. Users can share common components within all the tiers of the portal, ensuring consistency and leveraging investments that have been made in the production of existing EJB components. Tight XML integration through a flexible method of storing, searching and delivering XML to the user will provide true flexible and scalable portal applications.
- ASE 12.5 will support both event-based (a form of process-centric) and data-centric enterprise application integration with back-end systems including ERP, inventory and provisioning through SEEB, CIS/Omni and Replication Server products.
- The comprehensive security features in ASE 12.5 will ensure that all the transactions in the portal applications are safe and secure. The unstructured data management or content management using ASE will enable portal applications builders meet the requirements for conducting business over the Internet.

Just a Relational Database?

So, is ASE 12.5 still a relational database? Well, yes, it is. But my ex-director's comment that all RDBMSs are effectively the same was untrue then, and is even more so now. The new functionality that has been added to ASE does not detract from the capabilities that we have all come to rely on within ASE. Indeed, many of the features that have been added will benefit all users, whether portal user or not. However, for those enterprises who are looking to push the bounds of e-business, ASE 12.5 allows them to use class-leading performance and scalability, while providing functionality to assist users in creating e-business solutions that meet their customer requirements.

Ian Smart is Senior Technical Evangelist for Sybase's Enterprise Systems Group. He has addressed U.S. and European ISUG audiences on many occasions, as well as speaking at numerous user group and masterclass events in the UK. If you have a question you would like answered or a comment about an article, Ian can be reached at ian.smart@sybase.com.

In the Palm of Your Hand III

By Thomas Lamb

Hardware	Software	Operating System
Compaq Presario ACER Desktop AMD-K6 160MB	PowerBuilder 7.0 PowerBuilder 7.0	Windows 98 Windows 2000 Professional
DELL Latitude CPi D266XT 128MB	PowerBuilder 7.0	Windows NT 4.0 Workstation
Palm Vx	PocketC v3.8 CControls v1.0	Palm OS v3.5

One, two, three! Is that PocketC I see? Why, yes, it is! I hope that you've enjoyed this series of articles as much as I have enjoyed writing them. Because now we are shifting into high gear. Since the last column I've been busy changing jobs (twice), working on the Sybase TechWave 2000 Conference, and creating this year's *ISUG Conference and Journal CD*. If you haven't received it yet, it should be coming soon.

The objective of this series is to lead you through the mobilization of a PowerBuilder application to create a movie library application. In this column, we will work with a pseudo-compiler and native Palm OS databases. In the next, we will investigate the potential and capabilities of Sybase's UltraLite technology. To review the previous articles in this series, ISUG members can access the *ISUG Technical Journal* online at www.isug.com.

For reasons of brevity and clarity, not all of the code is listed. All of the source code discussed in this article can be found on my website at www.lamb.tj. For the other software mentioned, there are also links where available. For more information on Palm Computing in general, check out their web site at www.palm.com.

I would like to thank the folks at OrbWorks Software, developers of PocketC, for their help and support. For more information visit their website at www.orbworks.com. I also want to thank Mario Schlesinger, the developer of CControls.

nv_pocketCdb

The standard Palm database object introduced in the previous column has been extended to provide date support in the format YYYYMMDD. This format deviates from the standard Palm datetime format. Although the number for the date is stored in a four-byte integer field, it is stored separately from any time components. Additionally, PocketC provides a Date Library (PktCDateLib) extension that was developed to support standardized mechanisms for entering and performing calculations on dates.

Method	Arguments	Returns	Notes
Of_datetime	A_dt_value	Blob	Returns a null-padded four-byte blob representing the unsigned long value for the date in the format (year*1000)+(month*100)+day.
Of_date	A_dt_value	Blob	Returns a null-padded four-byte blob representing the unsigned long value for the date in the format (year*1000)+(month*100)+day.
Of_time	A_dt_value	Blob	Returns a null-padded four-byte blob representing the unsigned long value for the time in the format (hour*100)+minute.
Of_datetime	A_l_secs	Datetime	Returns a datetime value representing the number converted from (year*1000)+(month*100)+day.

Application Development Environment

Above, we define the platform(s) on which this work has been taking place. This information is provided for your reference, as no warranty is offered or implied that the code in this column will work on other platforms. If you find that it does not, please let me know by sending an e-mail to t.lamb@bigfoot.com.

Using PocketC

PocketC is a C-language based pseudo-compiler available for Palm devices. The main reason I chose to utilize this product was that it brings us closer to

our ultimate goal of a standard Palm application. Additionally, PocketC allowed me to do much of the development on the plane, in the train, sitting in front of the TV, etc. If you've ever used a product like Turbo Pascal, then you are pretty familiar with the way this product works.

Your source is entered as memos on your Palm device. An application is identified by the first line of the memo beginning with a double-slash (//) followed by the name of the application. Include files are identified in a similar fashion with a /* instead of the //.

For this application I've exported two tables/views to Palm databases from the application we've been working with. The first is the codes used to generate a series of popup windows for the user to pick from. To this, I've added an additional set of codes for boolean choices (i.e., Y or N). The other database provides the Movie table information.

In addition to PocketC, I'm also using CControls developed by Mario Schlesinger. CControls provides support for all of the Palm user interface controls, tables (including database linked tables), menus, dialog frames, and a very small footprint of include files. In addition to the controls, Mario has also developed CEditor. CEditor is a small application that actually allows you to build the user interface WYSIWYG. However, rather than go into details about either of these applications, we'll spend the time on the Compact Movie Database (CMDB) application.

Application Overview

First, PocketC can only deal with one open database at a time. With this in mind, the application has been architected to read the "Codes" first, while populating the popups. Once that has been done, the application is free to work and interface with the "Movie" database. Let's begin by reviewing the user interface elements I've chosen for this application. The primary UI will be a database table displaying the first two columns from the database (the movie name and whether it is in DVD format).

The user will be able to modify, add, and delete records. When modifying or adding records, a dialog box will be displayed with a table of the columns in the record. The first column in the record display will be the label of the column, and the second column will be the value currently in the database. As a dialog, the user will be able to cancel out without saving any changes to the database. If the user has asked to delete a record, then the title will be displayed in a confirmation dialog.

If the user has a choice of options during editing, these will be presented in a popup window.

Creating a PocketC Application

Let's begin by looking at the overall architecture of the application. Below is a table presenting the files (memos, etc) used by the application.

CMDB.c	The application file. This file creates the primary application controls and includes the primary event processing loop.
CMDB Create.c	An include file that creates the movie database if it is not present on the Palm device. This will also create an default version of the codes table with the column labels and the boolean choices.
CMDB Popup.c	An include file that manages the popup list information presented in the application.
CMDB Dialog.c	An include file that manages the add and modify procresses for records.
CControls.c	The primary CControls interface include file. This file automatically includes CControls1.c and CControls2.c
CControls1.c	
CControls2.c	
PktCDateLib	The PocketC Date management library.

I'll leave the investigation of the last four files for you. Let's take a look though at the other files, beginning with CMDB.c:

```
// CMDB.c
#define DBname      "CMDB"
#define DBfmt       "szszszszszszszszi4szi2szszszszsz"
#define DBtype      "sssssssssisisssss"
#define DBcols      16
#define DBnew       0
include "Ccontrols.c"
```

In this first part of the code, I've declared some C macros for the name of the primary database and the format of the data within a record. You'll notice there are two macros relating to the format of the data. The first (DBfmt) defines the actual format within the records themselves, and the second (DBtype) provides the data types for the record structure pointers used within the application. We'll look at this later in this application. The last line is an include statement for the CControls functionality to be "included" within our application.

```
// controls
Chandle ht1, hb[4];

// Declare and create the database
pointer DB;
include "CMDB Create.c"
include "CMDB Popup.c"
include "CMDB Dialog.c"
```

This section of code defines the controls used in the main view of the application. A Chandle is a special datatype created for CControls. DB, a pointer, is where the database record information will be stored while we work with it. The last three lines are *include files*. Due to the limitations on the size of memos on a Palm device, *include files* allow you to actually create programs that are large and robust.

```
initcontrols(){
 int i,r;

 // button 1
 hb[0]=Cbutton(1,147,35,0,1,4);
 Csetcontent(hb[0],"New");

 // button 2
 hb[1]=Cbutton(41,147,35,0,1,4);
 Csetcontent(hb[1],"Chg");

 // button 3
 hb[2]=Cbutton(81,147,35,0,1,4);
 Csetcontent(hb[2],"Del");

 // button 4
 hb[3]=Cbutton(121,147,35,0,1,4);
 Csetcontent(hb[3],"Quit");

 // create database-table
 ht1=CtableDB(1,20,158,10,2,DB, DBfmt);
 Csetrow(ht1,0,126,0);
 Csetrow(ht1,1,10,0);
}
```

The above function (*initcontrols*) creates the "objects" that provide the primary user interface for the application. The movie library is presented as a table (ht1) showing the first two columns of the database record. Additionally, there are four buttons providing an interface for requesting adds, changes, deletes, and quitting the application.

```
initDB(){
 DB = malloct(DBcols,DBtype);
 // create database
 if(!dbopen(DBname)) {
  dbdelete();
  createDB();
 }
}
```

The function *initDB* creates the pointer reference DB and attempts to open the database. If the database cannot be opened, the function *createDB* is invoked to create a general (empty) database:

```
drawscreen(){
 int i;
 clearg();
 title("CMovies Database");
 // draw controls
 for(i=0;i        <4;i++) Cdraw(hb[i]);
 Cdraw(ht1);
}
```

The drawscreen function actually draws the controls on the Palm device. The **clearg** function is a PocketC function that clears the graphics page. Taking a step back, PocketC provides two "pages" for interacting with users. The first page is the "text" page, and provides a convenient mechanism for displaying log information. The second page is the "graphics" page, providing the ability to draw controls (e.g., buttons, dialog frames, tables, etc.).

```
main() {
 int e;
 // initialize screen
 graph_on();
 title("Please wait....");

 // initialize popups
 initpopups();

 // open DB and init tableDB
 initDB();

 // initialize controls
 initcontrols();

 // display main-screen
 drawscreen();

 // message loop
 while(1){
  e=event(1);
  if (Cevent(hb[0],e))
   newrec();
  else if(Cevent(hb[1],e))
   modrec();
  else if (Cevent(hb[2],e))
   delrec();
  else if (Cevent(hb[3],e))
   break;
  else if(Cevent(ht1,e));
 }
 // close DB
 dbclose();
}
```

I've skipped over some of the functions in this code to get you to the main function and event loop of the application. The first thing we do is switch to the "graphics" page, by setting **graph_on()**. So that the user knows that something is happening during the initialization process, the title is set to "Please wait...". The functions *initpopups*, *initDB*, and *initcontrols* are invoked. The function *initpopups* is actually in one of the include files, and we'll get to it in a moment.

After the initialization process is complete, the screen is drawn and the main event loop begins. Since we have a "Quit" button, we'll do the loop forever, allowing the "Quit" button to break out of it. In the event loop, we wait for an event and then we'll check that event against each of the controls. Depending on the "button" pressed, we will either add, change, or delete a record:

```
modrec(){
 int i;
 i=Cgetcursel(ht1);
 dbrec(i);
 dbreadx(DB,DBfmt);
 if(ontable1(i)) {
  Csetfield(ht1,0,DB[0]);
  Csetfield(ht1,1,DB[1]);
  Cdraw(ht1);
 }
}
```

I'm leaving it to you to examine the *newrec* and *delrec* functions as your time permits. I did want to go through the *modrec* function so that you have a general understanding of what is involved. Since the CtableDB control provides a direct record link to the database, I'm using the *Cgetcursel* function to retrieve the index of the currently selected row within the control. This index corresponds directly with the record in the database. The PocketC functions *dbrec* and *dbreadx* position the application correctly in the database and then read the corresponding record into the DB record pointer, using the format DBfmt. The *ontable* function is then invoked, passing the record number. If the function returns a non-zero value (meaning the user pressed the OK button), the information is updated in the table and the table is then redrawn.

Maintaining Data in the Database

Let's examine the code for maintaining data in the database, found in the *ontable1* function of the CMDB Dialog.c include file.

```
int ontable1(int r){
```

This function accepts one parameter "r" as the record number within the database to be maintained. The function also returns an integer, corresponding to the user pressing the OK or Cancel buttons, allowing a boolean evaluation of results.

```
Chandle dhf1,dht1,dhb1,dhb2;
...
dhf1=Cframe(1,20,158,140);
Csetcontent(dhf1,DB[0]);
```

Just as we had to do in the main part of the application, we must define Chandle datatypes for each of the controls we will be using. In addition to the button and table styles controls used previously, we'll be using a frame (dbf1) control. The title of the frame is set to the Movie title, found in the first column of record (DB[0]).

```
dht1=Ctable(3,35,154,9,2);
Csetrow(dht1,0,40,0);
Csetrow(dht1,1,100,0);
for(k=0;k<DBcols;k++){
  Cadditem(dht1,slabels[k]);
  if (k==8) Csetfield(dht1,1, strdate(DB[k]));
  else Csetfield(dht1,1,DB[k]);
}
```

In the main part of the application, we used a CtableDB control to interact directly with the records of the database. For maintaining the information, I've chosen to use the Ctable control with two columns. The left column contains the column name of the field, and the right column displays the data.

```
...
 while(1) {
  e=event(1);
  if (Cevent(dhb1,e)) {
   dbrec(r);
   dbwritex(DB,DBfmt);
   irc=1;
   break;
  }
```

Skipping over the initialization of the buttons, and the drawing of the controls, we come to the event loop. If the user presses the OK button, the database is positioned to the correct record location, and the record is written out using the *dbwritex* function.

```
...
 else if (Cevent(dht1,e)) {
  di = Cgetcursel(dht1);
  sdb=Cgetfield(dht1,1);
  if (di==0){
   sdb=getsd( Cgetfield(dht1, 0), sdb);
   if(sdb=="") sdb=Cgetfield(dht1,1);
  } else if (((di>0) && (di<4)) || (di==9)) {
   if(Cpopupevent(dhpb)) sdb=Cgetcontent(dhpb);
  } else if (di==5) {
   if(Cpopupevent(dhpg)) sdb=Cgetcontent(dhpg);
  } else if (di==6) {
   if(Cpopupevent(dhpr)) sdb=Cgetcontent(dhpr);
  } else if (di == 8) {
   dt=datestr(sdb);
   dt = SelectFromDate(dt);
   if(dt>0){
    sdb = strdate(dt);
    DB[di] = dt;
   }
  }
  Csetfield(dht1,1,sdb);
  if(di!=8) DB[di] = sdb;
  Cdraw(dht1);
 }
```

Using the table as the interface, we check to see what row was selected (Cgetcursel). Then, based on the row selected, the application provides an appropriate interface. For ratings, genre, format, etc. one of the popups is used to allow the user a choice of items. For the date fields, the SelectFromDate function provided in the PktCDateLib library is used. Lastly, for string information (i.e., the title), the standard PocketC *getsd* function is used. Once the modification has taken place, the table is drawn again with the new information. The Cpopupevent displays a dropdown as a popup, waiting for the user to make a selection.

Initializing the Popup Controls

Let's take a look now at how the popups are created. The application uses a second database which contains records with three columns from the codes table in our PowerBuilder application. The first column groups the information, the second column merely acts as a counter, and the last column is the descriptive information that is placed in the primary database. Rather than creating the controls every time the user wants to maintain a record, and since PocketC only supports one open database at a time, the popup controls are initialized at the beginning of the application and already contain the available codes. However, since the application creates the maintenance table each time, the column labels are stored in an array.

```
dhpb=Cdropdown(41,46,20,2);
```

A popup is defined, in CControls, as a Cdropdown control. The control above provides the interface for the Boolean choices of Y or N.

```
if(DBcd[0]==1) {
 if (k<DBcols) slabels[k]=DBcd[2]; k++;
}
else if(DBcd[0]==2) Cadditem(dhpb,DBcd[2]);
else if(DBcd[0]==3) Cadditem(dhpg,DBcd[2]);
else if(DBcd[0]==4) Cadditem(dhpr,DBcd[2]);
else if(DBcd[0]==5) Cadditem(dhpbf,DBcd[2]);
else if(DBcd[0]==6) Cadditem(dhpf,DBcd[2]);
else if(DBcd[0]==7) Cadditem(dhps,DBcd[2]);
else if(DBcd[0]==8) Cadditem(dhpd,DBcd[2]);
```

The database for the codes is opened, and then looped through reading each record. The popup code (DB[0]) is evaluated to determine which popup the information will be added to. In the special case of the column labels, the information is stored in an array for the maintenance dialog.

Putting It All Together

There are two additional files that I've not mentioned yet. CMDB Create.c provides code to create an empty movie database and a "generic" codes database. As of the writing of this article, I'm also working on the CMDB Popup.c include file to allow the Palm application user to actually maintain the code groups that are used in the popups. All of the code used to build this PocketC Palm application can be found on my website.

Coming Up Next Time

In the next issue, I'll be taking this concept further, developing an actual Palm application again using a downloaded database extracted from a PowerBuilder Datawindow! Keep checking my website, as I will be posting the objects there as I work on them. ■

TechWave 2000 Turns on the Heat in Florida

By Cynthia Gill, ISUG Conference Director

"E-nabling your future" was the theme this year for the TechWave 2000 conference in Orlando, Florida, held on July 30–August 2. The opening session keynote set the pace with a smoke and laser spectacular that had actors performing on chairs, before Sybase CEO John Chen strode up to the stage and got down to e-business.

ISUG Achievement Award winners David Owen (left) and Wim ten Have (right) at the ISUG booth. Not pictured: Juergen Bittner and John Koontz.

The ISUG Achievement Award and IAD Awards

The keynote was capped by ISUG President Thom Lamb presenting the ISUG Awards for Outstanding Achievement, given each year to members who have contributed to Sybase's growth and awareness. Winners receive free conference passes, air and hotel accommodations.
For the year 2000, the awards were presented to:

David Owen, *Midsomer Consultants, Inc.*, who created new tools for ISUG including for installations, extractions, and process monitoring.
Wim ten Have, *Sybase*, who demonstrated significant expertise and dedication in his efforts to port Adaptive Server Enterprise to the Linux platform.
Juergen Bittner, *CEO and General Manager, SQL GmbH*, which was honored for implementing an Enterprise Application Server (EAServer) solution that uses the message-based sync mechanism of SAP.
John Koontz, *LifescapePro*, for implementation of an e-commerce system using Enterprise Application Server (EAServer) that ties together a wide variety of technologies and programming languages for the health care industry.

Maximum Technical Sessions

With more than 200 technical sessions taking place, TechWave featured a lot to learn about the latest in Sybase technology. Each year, ISUG performs a rigorous paper selection process in which papers are reviewed and ranked by a selection committee of volunteer user technical specialists. In return, ISUG and Sybase committee member receive a discount to the conference. This process helps to assure Sybase users good technical content in all the presentations.

Like last year, the afternoon sessions from the conference are available on the web for ISUG members at www.isug.com. In addition, attendees will be receiving a copy of the sessions on CD, compliments of the ISUG.

Exhibit Hall in the Spotlight

The conference Exhibit Hall, which had the largest contingent of exhibitors yet, was very popular. There were a lot of new products to learn about—with one featured by Ian Smart in his Enterprise Portals keynote session. The colored ISUG swizzle pens were a hit. Hope you got one. Congratulations to all the ISUG booth book winners!

ISUG Presidents Meet

The President's Luncheon, hosted by ISUG, provided an opportunity for User Group Presidents to network on common issues. Sybase's Pamela George, Vice President, Corporate Marketing; Raj Nathan, Vice President and General Manager of IAD; and Thom Lamb, ISUG President, spoke to the group. The presidents were very open about issues facing their groups.

Two Ask the Experts Sessions

Two "Ask the Expert/Enhancement" sessions were held on Wednesday afternoon, one for DBAs and one for developers. Users were given the opportunity to question Sybase product experts and find out about future enhancements.

Special Interest Groups Sessions

The Special Interest Groups gave Sybase users a forum to meet and learn about specific topics of interest relating to J2EE, EAServer, PowerDesigner, Wireless e-Business Applications, Mobile Solutions, ASE Server Management, Enterprise Portals, Continuous Availability, ASIQ, and Industry Warehouse Studios.

And Just Having Fun...

The Hospitality Suites sponsored by partners and Sybase produced a series of parties that everyone enjoyed, though the Sybase tent by the Swan seemed to be among the most popular. The winners on the quiz show seemed to enjoy themselves almost as much as the audience did—how could anyone get those questions wrong anyway? The IAD Big Rig event was also very popular: It's a must-see if it comes your way in the following months. On the last evening, the special event was a night at Walt Disney's Animal Kingdom—despite a little rain, it soon cleared up for a night of rides and Tarzan Rocks.

Check Out that Membership Value!

Be sure to visit our website at www.isug.com to get connected to Sybase and other users. By joining ISUG as a member for $75, you will receive a $75 conference discount off TechWave 2001 and many other benefits such as educational material and discounts, the TechWave conference CD, Sybase software, and a subscription to the *ISUG Technical Journal*.

Mark your calendars now for TechWave 2001 on August 12–16, in San Diego, California. Be proud, you are now part of a group of over 3,000 professionals from over 40 countries! ■

ISUG Board of Directors at TechWave 2000 (starting first row right): Cindy Bean, Thom Lamb, Joe Burger, Jeff Roberts, Luc Van der Veurst, Anthony Mandic, Dorus Kruse, Jay Hunt.
Not pictured: Cynthia Gill, Linda Morison, Kathy Ridley, Teresa Larson, Jaideep Chalwa, Jibu Abraham, Karen Pursch, Frank Monteverdi and David Johnstone.

Special Report

Sybase Goes to Thailand

By Anthony Mandic, Director, Asia-Pacific Region

The 2000 TechWave Asian User Conference was a hit at the Sheraton Laguna Phuket, held on October 21-22. The spectacular resort complex on Thailand's western resort island of Phuket, built around a chain of lagoons that many would have found reminiscent of Lake Buena Vista in Orlando, played host to 450 delegates, Sybase partners, and staff.

The delegates included a large contingent of PowerBuilder developers, but with representatives coming from India, Hong Kong, Korea, and Taiwan, there was something for everyone at this two-day event. The major themes were carried over from earlier TechWave events, in particular Sybase's enterprise portals. Sybase CEO and president, John Chen, impressed the audience in his keynote speech as he outlined Sybase's future directions for the Asian region.

After hours, well-planned activities included enjoying the best of local Thai and international cuisine and post-dinner shows at two of the largest restaurants in the region. The Sybase golf tournament was also well attended, despite the intermittent rain, with the beautiful course proving a great consolation for those who didn't win any prizes. ■

iAnywhere Supports Sybase Mobility

ISUG partner iAnywhere Solutions, a subsidiary of Sybase, has recently been spun off to provide mobile and wireless products and services to help organizations rapidly take part in the mobility revolution. iAnywhere Solutions' offerings include SQL Anywhere Studio, iAnywhere Wireless Server, and wireless hosting capabilities through its enterprise wireless ASP service.

SQL Anywhere Studio

SQL Anywhere Studio provides data management and enterprise synchronization, including Adaptive Server Anywhere. Its features include:

- Self-administering, self-tuning: Minimal DBA involvement is required. Graphical administration tool offers centralized administration of remote databases and the synchronization environment.
- Multi-platform support: Supports a wide range of operating systems, including Windows, Novell, UNIX and popular handheld and real-time platforms. Binary-compatible database files enable simple copying of files to multiple operating systems.
- Java and Internet support: Supports Java stored procedures and datatypes, and allows developers to create and store Java classes in the database to perform complex logic on the server. Enables the development and deployment of ultra-thin, database-driven Web applications.
- Optimized for handhelds and intelligent appliances: UltraLite deployment option enables organizations to deploy only the components of the database needed, resulting in a footprint as small as 50K. UltraLite leverages industry-standard SQL and provides referential integrity and transaction processing.
- Enterprise synchronization: SQL Anywhere includes MobiLink and SQL Remote for the secure, bi-directional synchronization of information between enterprise systems and remote devices. Remote devices connect via standard Internet protocols to the MobiLink synchronization server, which communicates via ODBC to heterogeneous datasources on the back end, including Sybase, Microsoft, Oracle and IBM. SQL Remote is based on a store-and-forward architecture that allows occasionally connected users to synchronize data using a file or message transfer mechanism such as FTP or e-mail. Only data changes are sent, minimizing communication costs.

iAnywhere Wireless Server

The iAnywhere Wireless Server enables access to applications, enterprise data, and Internet content through wireless devices.

- Fast, reliable access to information: Offline data storage with asynchronous messaging and data synchronization gives mobile workers the ability to work either on or offline.
- Customized content and distribution: Provides a framework to easily develop wireless applications. The framework exposes device-specific information, such as the content type supported by the browser, and provides the ability to define and access individual preferences.
- Security: Includes user authentication and authorization features as well as data encryption to address the security risks associated with remote access. Easy to integrate with existing security infrastructure such as firewalls, PKI technology, and virtual private networks (VPNs).
- Enterprise integration: Provides open enterprise integration into existing corporate systems including databases, enterprise resource planning systems, and legacy systems.
- Scalability and manageability: Designed to add users efficiently and effectively to handle significant growth. Provides built-in load balancing and failover to help distribute traffic dynamically among multiple systems, and is multi-threaded to enable performance tuning.

iAnywhere Solutions' enterprise wireless ASP service provides organizations with end-to-end, integrated wireless data solutions from system design and development to wireless hosting. iAnywhere Solutions can be found on the web at www.sybase.com/ianywhere. ■

INTERNATIONAL

Sybase User Group

ISUG Membership Application

ISUG Benefits

- Free Copy of SQL Anywhere Studio
- EA Server with PowerJ evaluation CD
- ASE 12.0 Technical Documentation CD
- A subscription to the quarterly ISUG Technical Journal
- Access to the online ISUG membership directory
- Access to the online ISUG Enhancement Request process
- Technical Training at local user groups
- TechWave Conference proceedings CD

Education Discounts

- Save 10% on Sybase education classes
- Save 20% on Certification Exams from Sylvan Prometric
- $75 discount on registration for TechWave and other ISUG-affiliated conferences

Note: Benefits are subject to change. Please refer to the ISUG website (www.isug.com) for current list.

please check one:

❑ individual membership*†	US	$ 75.00
❑ corporate membership*† (10 people)	US	$500.00
❑ government subscription*	US	$ 75.00

* Requires Sybase, Inc. Site ID or CBSS number.

† Members joining from Germany, the United Kingdom, Belgium, Luxembourg, the Netherlands, or France must join through their local user group. See LUG Directory for contact information.

Sybase Site ID or CBSS number (includes application & middleware product Site ID or CBSS number)

number _____

Your Site ID (CBSS number), found on your invoice or packing slip, is a 5-digit number followed by "- # -#".

please fill in the following contact information completely:**

name _____

title _____

phone _____

fax _____

email _____

company/organization _____

department _____

address _____

mailstop _____

city _____ state _____

postal code _____ country _____

If this is a corporate membership, please attach a separate sheet with the above information for nine additional persons.

Sybase products•

version platform

1. _____
2. _____
3. _____

•**Note:** *This information will be kept confidential.*

Are you a member of a Local User Group (LUG)? ❑ Yes ❑ No

Name of group _____

❑ Please send me information on the LUG in my area.

Are you a member of a Special Interest Group (SIG)? ❑ Yes ❑ No

name of group _____

I am most interested in the following SIGs:

- ❑ Adaptive Server
- ❑ Application Tools
- ❑ Architecture & Design
- ❑ Query & Reporting Tools
- ❑ Data Warehousing
- ❑ NT Server
- ❑ Systems Management
- ❑ VLDB
- ❑ WWW/Internet
- ❑ Middleware
- ❑ PowerBuilder

payment instructions

Please send a check made payable to "International Sybase User Group." Outside North America, please send a check for the currency equivalent.

Note: *All checks must be drawn from a US bank.*

To pay by credit card, please fill in the following information:

❑ VISA ❑ MASTERCARD

card number _____ exp. date _____

cardholder signature _____

Return this form with check or credit card information by enclosing in an envelope and applying stamp.

membership directory release form

Contact information will be distributed in the membership directory to ISUG members only. Product information will not be released. Please sign this form if you want your contact information published in the ISUG directory.

signature _____ date _____

non-disclosure agreement

ISUG members agree not to use any Sybase confidential information for any purpose except their business relationship with Sybase. Members also agree that they will not disclose confidential information to any person other than their company's employees who are directly involved in the use of Sybase products.

signature _____ date _____

Mail form to...

International Sybase User Group
6475 Christie Avenue
Emeryville, CA 94608
Fax: 510-922-0882
E-mail: isug@sybase.com

Note: Members joining from Germany, the United Kingdom, Belgium, Luxembourg, the Netherlands, or France must join through their local user group. See LUG Directory for contact information.

Board of Directors

President

Thomas J. Lamb
Chicago, IL
Phone: 815-621-5262
t.lamb@bigfoot.com

Vice President

Kathy Ridley
CGI
Houston, TX
Phone: 713-868-5537, x160
kridley@drthou.com

Secretary

Karen Pursch
Sybase
Boulder, CO
Phone: 720-820-0188
karenk@sybase.com

Treasurer

Luc Van der Veurst
Academic Hospital, VUB
Brussels, Belgium
Phone: 32-2-477-6980
lucv@az.vub.ac.be

Conference Director

Cynthia Gill
CGI
Long Beach, CA
Phone: 562-421-2070 x217
cynthia_gill@drthou.com

ISUG Technical Journal Director

Teresa A. Larson
VetCentric.com
Annapolis, MD
Phone: 410-571-6790
tlarson@vetcentric.com

Enhancements Co-Directors

Jibu Abraham
Jones Cyber Solutions
Denver, CO
Phone: 303-784-3612
jibu_abraham@hotmail.com

Jaideep Chawla
Pricewaterhouse Coopers
New York, NY
Phone: 703-645-5882
Jaideep.Chawla@us.pwcglobal.com

Membership Director

Joe Burger
The Longaberger Company
Newark, OH
Phone: 740-322-5091
jburger@longaberger.com

Australasia RUG Director

Anthony Mandic
Mandic Consulting Pty, Ltd.
Sydney, Australia
amandic@start.com.au

Electronic Media Director

Jeff Roberts
CCS Consulting
Atlanta, GA
Phone: 770-582-0360
Fax: 770-582-0590
jroberts@consultCCS.com

Members-at-Large

Linda Morison
Automated Data Sciences
Columbia, MD
Phone: 301-523-5360
Lmorison@erols.com

Peter F. Thawley
InterTrust
Santa Clara, CA
Phone: 408-855-0100
Fax: 408-222-6144
Peter@intertrust.com

Partner Membership Director

Frank Monteverdi
CGI
Houston, TX
Phone: 713-868-5537x149
Fax: 713-868-4014
Frank_monteverdi@drthou.com

European RUG Director

Dorus Kruse
Belastingdienst
Automatiseringscentrum
The Netherlands
Phone: 31 55 528 7945
doruskruse@hotmail.com

North American RUG Director

Cindy Bean
BMC Software, Inc.
Houston, TX
Phone: 713-918-1841
Fax: 713-918-1173
cynthia_bean@bmc.com

SIG Co-Directors

David Johnstone
Bank of Scotland
Edinburgh, Scotland
Phone: 44-0131-442-7957
david_johnstone@bankofscotland.co.uk

Jay Hunt
Automated Data Sciences
St. Louis, MO
Phone: 618-567-1177
Fax: 618-692-1188
djayhunt@charter.net

Sybase Contact:

Aline Martin
User Group Marketing
Sybase
Emeryville, CA
Phone: 510-922-7525
Fax: 510-922-0882
amartin@sybase.com

Sybase User Group Directory

European Region

Austria
Dr. Wolfgang Lipa
ZAMG
Phone: 43-0222-36-44-53 x 2603
Fax: 43-0222-369-1233

Belgium "Blues"
Luc Van der Veurst
Academic Hospital VUB
Phone: 32-2-477-6980
lucv@az.vub.ac.be

Denmark
Lars Henrichsen
COWI Consult A/S
Phone: 45-45-972068
or 45-45-972211
Fax: 45-45-972212
lh@cowi.dk

Estonia
Gustav Seier
Tallinn, Estonia
Phone: 372 6655467
gustav.seier@eyp.ec

Finland
Kristiina Salminen
Sanoma Corporation
Phone: 358 9 122 87 3184
kristiina.salminen@sanoma.fi

France "Fibonacci"
M. Gerard Lledo
C.E.A.
CENTRE ETUDES SACLAY
Phone: 33-1-6908-9616
Fax: 33-1-6908-9608

Germany

Sybase User Group
Jens Timmermann
TOPOLOGIX GmbH
Phone: 49-40-352-253
Fax: 49-40-340-340
jens.timmermann@topologix.com

PowerBuilder User Group
Ludwin Feiten
Power People
Vorstand@pbugg.de
www.pbugg.de

Ireland
David Quirke
Phone: 00 353 1 677 6777
Fax: 00 353 1 6776614
dquirke@sybase.ie

The Netherlands
Jolanda Zwaan
Phone: 036 5226249
Fax: 036 5226249
dsug@tref.nl

Norway
Irene Negarden
Norsk Hydro ASA
Tel: +47-22738987
Irene.Negarden@hydro.com

Spain
Manuel Blanco Ulled
Unipapel, S.A.
Phone: 91-806-96-10
Fax: 91-803-52-22
mblanco@unipapel.com

Switzerland
Dr. Roberto Buzzi
Zurich Versicherungs
Phone: 41-1-205-2121
Fax: 41-1-205-3375
chzurkrb@ibmmail.com

Turkey
Levent Sensezgin
Sybase Turkey
Phone: 90-212-284-8339
Fax: 90-212-284-8342
levent.sensezgin@sybase.com.tr

United Kingdom
Khosro Parnian
Sensitek, Ltd.
Phone: +44-0171-325-0374
parnian_khosro@jpmorgan.com

UK PowerBuilder User Group
Anne Bocock
PB Associates
Phone: 020-84213533
Fax: 020-84201420
anne@pbug.co.uk

International

Argentina
Sergio Di Cuffa
Sybase Argentina
Phone: 54-1-393-0421
Fax: 54-1-326-7039

Australia
Igor Geninson
IBS Technology
Phone: 61-02-388-7675
Fax: 61-02-388-8067

Brazil
Luiz Guilherme Mendonca
LOCUS Ltd.
Phone: 55-21-553-3086
Fax: 55-21-551-8461
lguilherme@locus.com.br

Chile
Jean Pierre Lefranc
Sybase Chile
Phone: 56-2-3306700
Fax: 56-2-3306800

New Zealand
Shayne Duncan
Sybase NZ, Ltd.
Phone: 64-4-473-3661
Fax: 64-4-499-9068
shayne@sybase.com

Saudi Arabia
A. Omran
Al-Omran
Phone: 96-61-4662373
Fax: 96-61-4662502

South Africa
Deirdre Martin
c367455@sn2.edsa.co.za

Thailand
Chakkrapong Tongsak
Bangkok University
Fax: 65-273-9159
ct@lily.bu.ac.th

United Arab Emirates
George Khouri
Sybase Products - Middle East
Abu Dhabi
Phone: 97-1-2-325-911
Fax: 97-1-2-340-850

Sybase Local User Groups of North America

Northeast Region

Regional Contact
Jan Barcelou
Barings Asset Management
Phone: 617-946-5325
Fax: 617-946-5410
Jan.barcelou@baring-asset.com

Albany, New York PowerBuilder User Group
Greg Fisher
gfisher@dttus.com
www.anypbug.org

Boston PowerBuilder User Group
Olga Demidova
SOFTRAX Corp.
ODemidova@softrax.com
www.middlewarecg.com/pbug

Boston (SNEKUS–Sybase North East Kingdom Users Society)
Bruce Driver, Consultant
Phone: 508-443-3310
bdriver@btrgroup.com or
bdriver@csi.com
www.snekus.org

Harrisburg, PA, PowerBuilder User Group
Dennis Rehm
dennisrehm@appliedcomputing.net

New Jersey
Bob Munson
Phone: 973-367-3634
robert.munson@prudential.com

New Jersey PowerBuilder User Group
Boris Gasin
Phone: 973-402-5600
bgasin@dynamictechgroup.com
www.njpbug.org

Ontario Sybase User Group
Chris Baker, Visual Systems Development Group
Phone: 416-591-0005 x245
OSUG@interlog.com
www.interlog.com/~osug

Ottawa
Tony Antonallo
Phone: 613-798-7507

Ottawa PowerBuilder User Group
Carole Hargrave
Saphire Consulting Services, Inc.
Phone: 613-292-2018
opbug@opbug.com
www.opbug.com

Philadelphia
Sybase, Inc.
Phone: 610-260-4300
Fax: 610-260-4399

Philadelphia Sybase IAD User Group
Chuck Miller
cmiller@appltech.net
pbug.fcg.com/PBUG/index.html

Toronto PowerBuilder User Group
Paul Bis
Tpbug@interlog.com
www.interlog.com/~tpbug

Southeast Region

Atlanta PowerBuilder User Group
Jeff Roberts
CCS Consulting
Phone: 770-582-0360
Fax: 770-582-0590
jroberts@consultccs.com
www.apbug.org

Baltimore Area PowerBuilder User Group
Debbie Arczynski
softquip@home.com
www.geocities.com/bapbug

Baltimore/Washington, D.C.
Keith Altman
Dcasug@dgsys.com
www2.dgsys.com/~dcasug

Birmingham PowerBuilder User Group
Dennis Marcum
Dennis_marcum@protective.com
www.highland-consulting.com/bpbug/index.html

Boca Raton
John Glover
Data Breeze
Phone: 954-427-4416 x309
Fax: 954-427-0280
jdg@phamis.com

Charlotte PowerBuilder User Group
Alan Augustine
aaugustine@ttpartners.com
www.charpbug.org

Charlotte Sybase User Group
Bob Kunkle
Phone: 704-375-5788

Greensboro/Piedmont Triad PowerBuilder User Group
James Stoertz
jstoertz@fitech-gso.com

Louisiana PowerBuilder User Group
Joe Sherrill
joesherrill@msn.com
www.notrs.com/lapbug

Miami
Ray Liera
University of Miami
Phone: 305-284-4207
Fax: 305-284-4753
Rleira@miami.edu

Music City Powersoft User Group
Jeff Gibson
jgibson@frontrunner.com
www.frontrunner.com/mcpug

Nashville
Jimmy Hogan
Phone: 615-641-5550
Fax: 615-841-5556

North Carolina
Clay Bullard
SMCI
Phone: 704-375-5788
Fax: 704-375-5699
smcic@ix.netcom.com

Northwest Arkansas Sybase User Group
Dan Luttrell
vpbo@vicsinc.com
www.cyberspace.org/~nwapbug

Orlando PowerBuilder User Group
Bryan Enochs
AT&T
Phone: 407-858-8320
Fax: 407-858-8352
benochs@ao.net
www.wedowebs.com/opbug

Tampa Bay Area
Jeffrey Garbus
Phone: 813-949-7016
Jeffg@soaringeagleltd.com
www.soaringeagleltd.com/sybase_local_user_group.htm

Washington Area Sybase Enterprise Application Tools User Group (WA/SEATUG)
Theo Rushin, Jr.
National Institutes of Health
Phone: 301-402-4609
Fax: 301-480-6105
rushint@mail.nih.gov

North Central Region

Regional Contact
Gaynel Walden
AT&T
Phone: 816-995-3723
gaynel@att.com

Central Ohio PowerBuilder User Group
Barry McDonald
bmcdonald@CBSInc.com
www.cmnpbug.com

Chicago Sybase User Group
John Da Silva
Power 2000
Phone: 630-369-7175
Fax: 630-369-8042
jdasilva@power2000.com
www.csug.com

ChicagoLand Sybase Tools User Group
Michael Baraz
Phone: 630-235-4529
mbaraz@tradesolutions.com
www.cpbug.org

Cincinnati/Dayton
Scott Stegman
Phone: 513-241-5949
Fax: 513-241-6731
sstegman@clientserver.com

Des Moines PowerBuilder User Group
Mary Eagan
meagan@ifmc.org

East Michigan PowerBuilder User Group
Benny Cheung
benny@mpbug.org
www.mpbug.org

Indianapolis
Julie Clark
Phone: 317-501-7559
jclark@moser-inc.com
www.cs.bsu.edu/homepages/sam/isug

Kansas City Area Sybase User Group
George Meiers
Hallmark Cards, Inc.
Phone: 816-545-6395
Gmeier1@hallmark.com

Michigan PowerBuilder User Group
Jagdish Karira
Phone: 248-524-3280
jagdish@techie.com

Minneapolis, MN
Maendra Saraswate
Hunter Software
Phone: 612-943-3986
Fax: 612-941-0933
msarasw@primenet.com

Minneapolis/Twin Cities Sybase Tools User Group
Bob Sharp
Phone: 612-483-1506
www.tcpbug.com

PowerBuilder User Group of Northeast Ohio
Todd Senauskas
tsenauskas@datavantage.com
www.internet-club.com/usa/neopbug

St. Louis Sybase Internet Tools User Group
Dave Blankenship
president@stlpbug.org
www.stlpbug.org

Winnipeg
Jim Novisat
Great–West Life Assurance Co.
Phone: 204-946-7748
Fax: 204-946-4567
jzn@gwl.ca

Wisconsin Sybase User Group
Michelle Murphy
Wisconsin Electric Power Co.
Phone: 414-221-2068
Fax: 414-221-4744
michelle.murphy@wepco.com

Bill Mitchell
Northwestern Mutual Life
Phone: 414-299-4022
Fax: 414-299-1686
Billmitchell@northwesternmutual.com
www.reveregroup.com/wisug

SW-Mountain Region

Regional Contact
Cindy Bean
BMC Software, Inc.
Phone: 713-918-1841
Fax: 713-918-1173
cynthia_bean@bmc.com

Austin
Rob Gustavson
BCS Systems, Inc.
Phone: 512-921-0074
Rgustavson@rfdinc.com

Colorado PowerBuilder User Group
Mark Juodawlkis
mjuodawlkis@gr.com
www.co-pbug.org

Colorado Springs PowerBuilder User Group
Judith McEntee
jmcentee@gr.com
www.camelotconsulting.com/cspbug

Dallas Sybase User Group
Phil Adams
FireSteed Software, Inc.
Phone: 817-296-6238
Fax: 817-282-8980
pcadams@firesteedsoftware.com
www.firesteedsoftware.com/tsug.html

Dallas/Ft. Worth PowerBuilder User Group
Marcie Jones
marciejones@yahoo.com
www.dfwpbug.com

Houston (Sybase Users' Group of Texas)
Tony Broussard
TradeCapture.com
Phone: 713-752-4063
admin@sugtx.org
www.sugtx.org

Inter-Mountain/Utah
Dianne Garcia
Discover Brokerage Direct
Phone: 801-902-4222
Garcia@discoverbrokerage.com

Oklahoma City PowerBuilder User Group
Dianna Demotto
okcpbug@usa.net
okcpbug.iwarp.com

Rocky Mountain/Denver/ Salt Lake City
Doug Thomas
Time Warner Telecom
Phone: 303-566-1432
Fax: 303-566-1400
dougtho@geocities.com
rmsug.intranets.com

San Antonio Sybase User Group
Bill Purnell
purnell@texas.net

San Antonio PowerBuilder User Group
Richard Carrier
rcarrier@tesoropetroleum.com
www.pfccheatsheet.com/pbugsa.htl

Tulsa PowerBuilder User Group
Mike Deasy
mdeasy@energy.twc.com
www.geocities.com/tulsa_pbug

Wichita Kansas PowerBuilder User Group
Rodney Hughes
Phone: 316-517-1847
rjhughes@cessna.textron.com

Far West Region

Regional Contact
Cynthia Gill, CGI
Phone: 562-421-2070 x217
cynthia.gill@cgiusa.com

Arizona Sybase/Powersoft User Group
John Lewis
oberon@home.com
azspug.webjump.com

Hawaii
Sterling Yee
Hawaiian Electronic Industries
Phone: 808-532-5870
Fax: 808-532-5828
syee@hei.com

Los Angeles
Cory Isaacson, Compuflex
Phone: 818-772-7990
Fax: 818-772-7999

Los Angles PowerBuilder User Group
Brian J. Smith
Phone: 310-221-0798
75530.332@compuserve.com

Orange County PowerBuilder User Group
Victor A. Reinhart
victora.reinhart@phs.com
www.ocpbug.com

Riverside Area Poweruilder User Group
Joe Fontaine
Riverside Cty. Dept. Information Technology
Phone: 909-955-3692
Fax: 909-955-9390
jfontain@pe.net
www.pe.net/!jfontain/rapbug.html

San Diego PowerBuilder User Group
Joe Hermiz
Phone: 619-334-7638
joe@workcompsoftware.com

Northwest Region

Regional Contact
Matt Townsend, Ayupp Inc.
Phone: 707-829-1173
Matt_townsend@earthlink.net

Boise
Michelle Featherston
Micron Semiconductor
Phone: 208-368-4498
Fax: 208-368-1043
mfeatherston@micron.com

Calgary/Edmonton
David Owen
dowen@trimac.com

Edmonton PowerBuilder User Group
Theodore J. Allen
73614.3174@compuserve.com
www.freenet.edmonton.ab.ca/~vschmid/pbug.htm

Portland, Oregon Sybase Developers User Group
Brad Ashton
bashton@centric-corp.com
www.teleport.com/~wagnerc/pbug

Pacific Northwest/Seattle (SNUG)
Chris Young
Cascade Software, Inc.
Phone: 206-224-3725
Fax: 206-224-6205
cyoung@cascadia-sw.com
www.cascadia-sw.com/snug

San Francisco PowerBuilder User Group
Amine Khechfe
Phone: 650-321-2640 x104
a@psi-systems.com
www.sfpbug.org/

Seattle/Northwest PowerBuilder User Group
Curtis Hanner
curtis@hanner.com
www.cascadia-sw.com/nwpbug

Vancouver PowerBuilder User Group
Dennis Lee
Phone: 604-893-7040
dlee@vusual.com

Get Ready—Get Set...

ISUG Is Coming To You!

Get ready for the ISUG Innovation Tour 2001 roadshows—a dazzling display of technology and user information coming to a European location near you. Prepare to expand your knowledge base, improve your computing environment, and network with your ISUG colleagues!

See

Presentations on enterprise portal database and Internet development and deployment products

Learn

About ASE 12.5, enterprise portal products, and PowerBuilder

For

Database administrators and developers

Register

Online at the ISUG website at www.isug.com

We're Coming to You in 2001!

Stockholm, Sweden	Feb. 26
Oslo, Norway	Feb. 28
London, United Kingdom	March 2
Brussels, Belgium	March 5
Düsseldorf, Germany	March 7
Paris, France	Date TBA
Dublin, Ireland (database)	March 12
Zurich, Switz. (developers)	March 12