ISUG TECHNICAL JOURNAL

SECOND QUARTER 1999

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

INSIDE

Administrating Sybase Interfaces Files on the Web

Global Peer-to-Peer Data Replication Using Sybase Replication Server

Programming Java Inside Sybase's Adaptive Server Databases

How to Set ASE Login Passwords (Back) to Blank

INTERNATIONAL

Sybase User Group

PRESIDENT'S MESSAGE

Dear ISUG Members,

This year, for the first time, the ISUG North American Conference, the Powersoft Conference, the Middleware Technical Summit, and the European ISUG Conference are being combined into one major action-packed event—Sybase TechWave '99. For our regular ISUG conference attendees, please note that the ISUG is working to keep the conference as technical as possible and maintain your favorite sessions and activities.

If you haven't received an agenda for TechWave '99 yet, I am sure that it will soon be dropped into your mailbox. The agenda and Registration forms are also available through the ISUG website—just follow the TechWave link from our home page. I would like to personally invite you all to attend this conference, and especially encourage my European colleagues to cross the ocean to participate in the largest user conference to ever take place.

The ISUG board of directors continues to be active in other areas. We are working on developing our relationships with Sybase's four new divisions. During our May board meeting in Concord, MA, for example, we met with Sybase's Internet Application division, to ensure that ISUG continues to support the interests of all Sybase users.

We are also continuing to grow our ISUG Partners program, and would like to welcome Autometric as a new partner company (see profile, page 29). We are also especially pleased to welcome Dorus Kruse to the board, filling the position of European RUG Director.

The 1999 Asia Pacific User Conference, held in Sydney Australia, was also a big success. This regional user group has seen an outstanding 30% growth rate just during the past six months and many conference sessions were standing room only. For a full conference report, see page 30.

We are currently updating the ISUG website at www.isug.com, every five to six weeks. Here you will find new information on conferences, new technical data and links, the latest reports and calendars of user group activities, online version of the *ISUG Technical Journal* articles, and the most up-to-date user contact information. If you are interested in launching a local user group in your area, please access the new section of the site that contains information and a Powerpoint presentation on how to start up and support your LUG.

If you are an ISUG member and haven't yet received your userid and password to gain access to the private section of our website at www.isug.com, please go to the website, click on the Forgot Password? button, and enter the same email address you submitted on your application form to receive your password via email. You can also keep your mailing address current by filling out the Update Personal Info form in the private section of our website. If you need further assistance, please send an email to info@isug.com.

Sincerely,
Luc Van der Veurst
ISUG President
Akademisch Ziekenhuis VUB
Brussels, Belgium

Table of Contents

FEATURES

Administrating Sybase Interfaces Files on the Web 2
By Zhengyu Wang and Joseph Dunn

Global Peer-to-Peer Data Replication Using Sybase Replication Server 9
By Mich Talebzadeh

Programming Java Inside Sybase's Adaptive Server Databases 15
By Sam Blanchard

How to Set ASE Login Passwords (Back) to Blank 28
By Rob Verschoor

Autometric's Spatial Query Server — Spatial Data Management for Sybase Users 29
An ISUG partner profile

Asia-Pacific Conference Draws New Members 30

DEPARTMENTS

Server Views *Peter Thawley* 20

PowerBuilder Tips & Techniques 25
Thomas Lamb

News Brief 31

Membership Report 32

READER INFORMATION

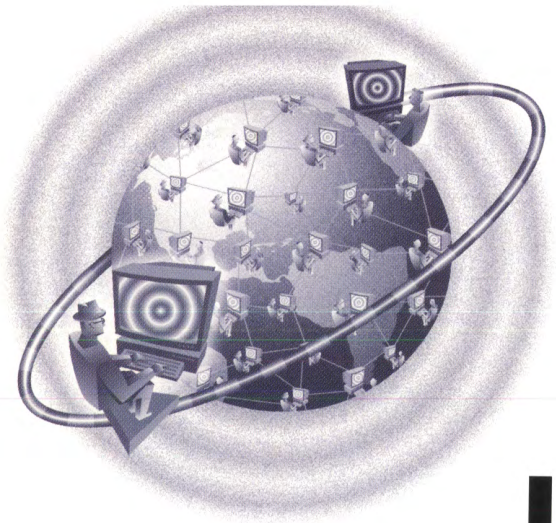
Calendar of Upcoming Events 33

ISUG Membership Application 34

ISUG Board Directory 35

Sybase User Group Directory 36

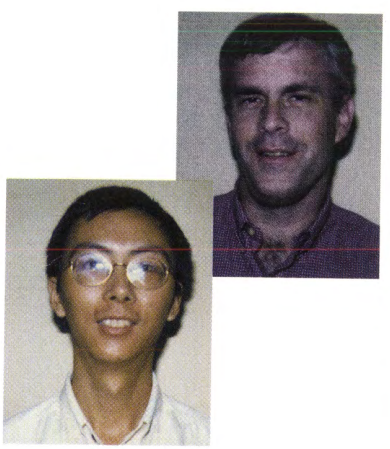
ISUG Technical Journal



Administrating Sybase Interfaces Files on the Web

By Zhengyu Wang and Joseph Dunn

Presenting new techniques for modifying and distributing your network's interfaces information in today's distributed environments.



Zhengyu Wang (bottom) is a senior consultant for Sybase, and has more than six years of experience in software development, system administration, and performance analysis. Joseph Dunn is a manager of database operations at AOL, with 18 years in telecommunications and finance services including six years on Sybase products. They can be reached at: Zhengyu.Wang@sybase.com

In a client-server computing environment, clients communicate with one or more Sybase Adaptive Servers. Servers communicate with other Adaptive Servers and Open Server applications on a network via remote procedure calls. In order for clients and servers to interact with one another, each needs to know where the other resides on the network.

Sybase stores this network service information in the interfaces file, listing the name and address of every known server. In a heterogeneous environment, the name, location, and format of the interfaces file differ based on its resident operating system. By default, the file is called *interfaces* on UNIX and *sql.ini* on the PC platform.

The format of Adaptive Server addresses in the interfaces files differ between network protocols. The typical format, which applies to most UNIX platforms (HP-UX, IBM RS/6000, SGI, etc.), is given below. Though Sun Solaris, Digital Unix and Windows platforms require different formats of interfaces files, it is simple to convert them.

```
MYSERVER
    query tcp ether myserver.sybdb.com 2550
    master tcp ether myserver.sybdb.com 2550
```

MYSERVER is the name of Adaptive Server. Each server entry contains service type, protocol, network, machine, and port respectively. Service type *query* is used by clients to find Adaptive Server while *master* is used by server to determine the port it should "listen" on.

As a system administrator, it is your responsibility to modify the interfaces file and distribute it throughout the environment so that client-to-server and server-to-server communication can take place. In a large distributed client-server computing environment, there are hundreds of Sybase Adaptive Servers, Monitor servers, legacy servers, IQ servers, XP servers, Backup and Open Server applications which are connected to by many clients. The interfaces file may contain hundreds of server entries and reside on thousands of locations over the network. These interfaces files may also be in different formats dependent on platforms in a heterogeneous computing environment.

In the meantime, to support computing environment growth, the interfaces files are changed on a frequent basis as new servers are put into production, existing servers are moved to new hosts, old servers are phased out, or host addresses are changed. After making the change, system administrators must keep track of the portion of the computing environment which will be affected by the change and then distribute the interfaces file to appropriate clients and servers.

All this adds up to the fact that the risk of distributing a bad interfaces file can be enormous. We have seen cases where clients intended to communicate with production server A. However, wrong interfaces entries for server A made clients communicate with server B. It may potentially destroy critical information stored in server B, and in the worst scenarios, bring down business applications relying on servers A and B.

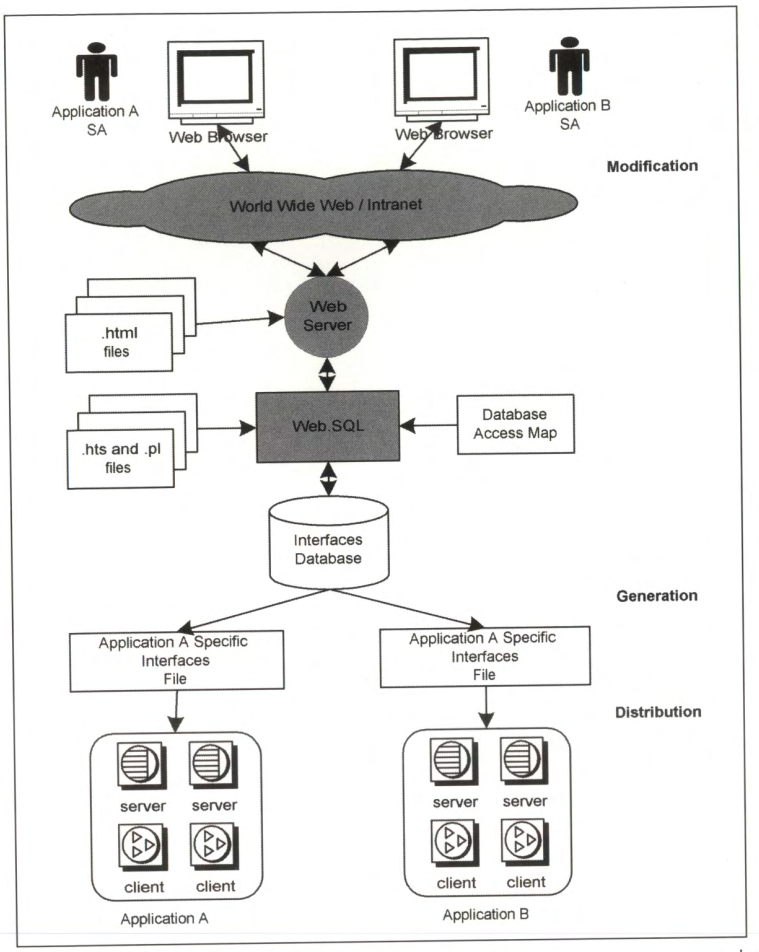


Figure 1. Web-based Interfaces File Administration

To administrate Sybase interfaces files in our environment and to relieve the burden of system administrators, we evolved into an automated web-based system where server entries can be added and modified using a web front-end, while system-specific interfaces files can be generated, verified and finally distributed to the correct locations. Users manipulate interfaces entries via web pages on their web browsers. By simply clicking the "submit" button, data modification requests are sent to the web server via our corporate intranet.

Architecture

The basic premise of our web-based interfaces file administration system is to store interfaces file entries in a database and use a web front-end to add, modify, delete, and search server entries. After modification in the database is made through web browsers, application-specific interfaces files are generated dynamically and scripts to distribute the interfaces files out to each location are generated. Scripts to test the interfaces files for accuracy are also generated and executed before the files are converted to different formats and distributed to servers and clients. Figure 1 shows the system architecture.

Security is always a concern when building a web-based system. Only system administrators are authorized to modify and view interfaces entries on our web pages. As with most secure web pages on the Internet, ours are also password-protected. Users must give the correct name and password to access all web pages in the system. We are also planning to use a secure web server in the future.

The technologies to build the system include Sybase web.sql, HTML, CGI/Perl, Sybperl, and JavaScript.

- The web front-end pages were written in HTML version 3.0 and JavaScript version 1.1.
- The middleware is composed of Apache Web Server version 1.2.4 and web.sql version 1.2, running on SGI IRIX 6.2 operating systems.
- The database, storing the interfaces file entries, resides on Sybase Adaptive Server 11.0.3.3, running on SGI IRIX 6.4 operating systems.
- The backend scripts to generate and distribute interfaces files were written in Perl version 5.004 and Sybperl version 2.009.

These technologies were chosen for their ability to work together seamlessly. However, the system is not restricted to the hardware and software mentioned above. Readers can develop the system on their chosen version of Sybase Adaptive Server and Web Server. Web pages and associated programs can be developed in any manner, as long as they provide a front-end for users to search and modify server entries in the database.

Setting Up the Interface File Database

The structure of the database which stores interfaces file server entries is very simple. There is only one table, with the following schema:

```
server      varchar(30)    # server name
srvr_type   varchar(10)    # server type
query       varchar(10)    # query/master
rownum      smallint       # row order
protocol    varchar(10)    # protocol
network     varchar(10)    # network
hostname    varchar(50)    # host name
port        smallint       # port number
subsystem   varchar(10)    # subsystem
dept        varchar(20)    # department name
vendor      varchar(20)    # database vendor
```

The *server*, *query*, *protocol*, *network*, *hostname*, and *port* are required elements in the interfaces file, therefore are required in the table. *srvr_type* is added to differentiate server types, such as Adaptive Server, Backup Server, Replication Server, Adaptive IQ Server, LTM (Log Transfer Manager for Replication Server), etc.

It also contains entries for other database vendors. Even though they are not part of the interfaces file, it provides a mechanism of documenting all database servers in our enterprise environment. Interfaces files used by clients on the network generally do not contain Backup Server, Replication Server, etc., so their interfaces files may only contain server entries for Adaptive Server and IQ, which further reduces the file sizes.

rownum indicates the order in which entries for the same server are placed in the interfaces files. *subsystem* is used to categorize each server entry according to the application the server supports. Note that in our environment, one server usually supports one application. Servers supporting multiple applications (such as test or development servers) are categorized as *generic* as their subsystem value, to indicate that these server entries must exist in each interfaces file. Though not an ideal solution, it keeps our table structure in the database simple. To differentiate servers that support multiple applications, however, it is necessary to create two more tables to implement many-to-many relationships.

Using the values in the *subsystem* and *hostname* columns, we can determine which hosts each interfaces file is pushed to. *dept* identifies which corporate department administers the database server. At the highest level, this provides the first level of separation of the interfaces files.

The System Front-End

The front end of our system is built on a web browser for easy access from PC or UNIX platforms over our corporate intranet. System administrators can add, delete, search, update, and preview interfaces entries and files on the web browser by simply clicking on the linked URL. When a URL is selected, the http server translates the URL into a pathname for a file on the server's host machine. If the file is an HTML, GIF, JPEG, or other file type that the web server understands, the http server sends the file directly to the browser. If the file is a program residing in an authorized directory, the web server executes the program according to the Common Gateway Interface (CGI) and sends the output of the program to the web browser.

web.sql runs as a CGI program in our system in order to return dynamic HTML pages to the browser with the help of the HyperTextSybase (HTS) file. The HTS file format is an extension of the standard HTML format. It supports one additional tag, **<SYB>**, allowing Transact-SQL and Perl statements to be embedded in HTML files. When a browser requests a URL that translates to an HTS file, the http server passes the request to the web.sql program. The web.sql program reads the specified HTS file, processes the database and Perl requests contained in that file, and composes HTML output for the http server to pass to the browser.

In order to establish database connections, maintain security of HTS files, and prevent unauthorized database access, web.sql maintains a database access map. The access map defines valid default database connections and associations for specific HTS files or directories. Entries in the map specify the server, login, and encrypted password to use for an HTS file or a group of HTS files. The access map can be created and updated with the Sybase web.sql Administration web page using a browser.

Though a CGI/PERL program can establish database connections and manipulate data without relying on web.sql, there are some advantages to using it. An HTS file is associated with a database connection so that the server, login, and password are transparent to developers and users. Access to the database server is allowed only by using our web pages. To maintain a high level of security, logins and passwords are changed frequently, but users are insulated from these changes. The underlying database can also be moved from one server to another without the users needing to know the new location. Any of these changes can be made without

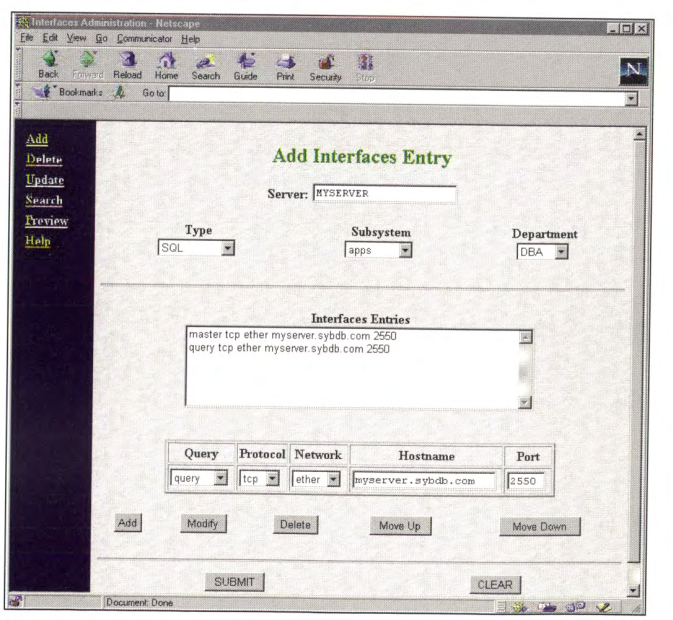


Figure 2. Add Interfaces Entry Web Page

requiring modification of our HTS files. Only the database connections associated with the files have to be adjusted on the web.sql administration page—which simplifies the administration of the website significantly.

Manipulating Interface Entries from the Browser

We would like to show some of our web pages to give readers a feeling of how interfaces entries stored in the database can be manipulated on the browser. The first web page is used to add interfaces entries (as shown in Figure 2). The server name is entered in a text field while server type, subsystem, and department values can be selected from the pull-down list.

In our environment, servers are named according to certain naming conventions. For example, a Backup Server associated with an Adaptive Server is named by appending suffix **_BS** to the name of Adaptive Server. Monitor Server is named by appending suffix **_MS**. By following these conventions, recommended by Sybase, server types can be recognized without requiring users to select the value from the list. It significantly reduces human error when selecting a corresponding server type. Each interfaces entry associated with the same server can be added, modified, and deleted individually. Users can adjust the query and master lines by moving them up and down in the window. The way these lines appear in the window is how they will be placed in the interfaces file.

After the "Submit" button is clicked, each required field is verified before the interfaces entries are sent to the web server. The web.sql program connects to the database and verifies whether the server already exists and whether the server is trying to use the same port as an existing server on the same host. Correct server entries are added into the database while incorrect entries are returned to the user with error messages to the web browser. To prevent incorrect entries from being entered into the database and to maintain consistency of database contents, each server entry is verified individually before sending them to the web server and then verified against other server entries before being added to the database. This double verification technique is also used in the "Delete" and "Update" web pages.

The "Search" web page is also worth mentioning (as shown in Figure 3). Users can input any patterns into the server and host fields and select multiple values in type, subsystem, and department list based on their search criteria. Users' selection criteria are sent to the web server. The web server runs the web.sql script, creating a complex SQL select statement to query the database. The results are formatted and returned to the web browser to display the results.

When a user requests to update or delete interfaces entries, the same search functionality is used to find the records meeting the users' deletion or update criteria. After the entries are selected for delete or update, each field is displayed on the browser similar to the "Add" page. Users then modify any field and send the request to the web server which performs the deletion and update tasks. When entries are chosen to be deleted, the user is asked to verify they want the records deleted before sending the request to the web server.

As we mentioned earlier, interfaces file generation and distribution are handled by independent scripts. However, interfaces files can be previewed on the "Preview" web page, which shows application-specific or global interfaces files in different formats. Interfaces files are generated by a CGI Perl program and displayed on the browser as a text filetype. Apart from previewing the interfaces file to ensure accuracy, users can save the file on the machine where the browser is running.

Though it may violate our centralized interfaces file distribution mechanism, this is very useful in some instances. For example, if a system administrator is using a machine which is not in our enterprise environment (personal computer at home or laptop on the road), his machine may not be in our distribution list of hosts. This machine occasionally connects to the corporate network and needs an up-to-date interfaces file to establish connections to servers, which can be saved via the "Preview" function. This is also a "Pull" method to distribute interfaces files, which is discussed on the next page.

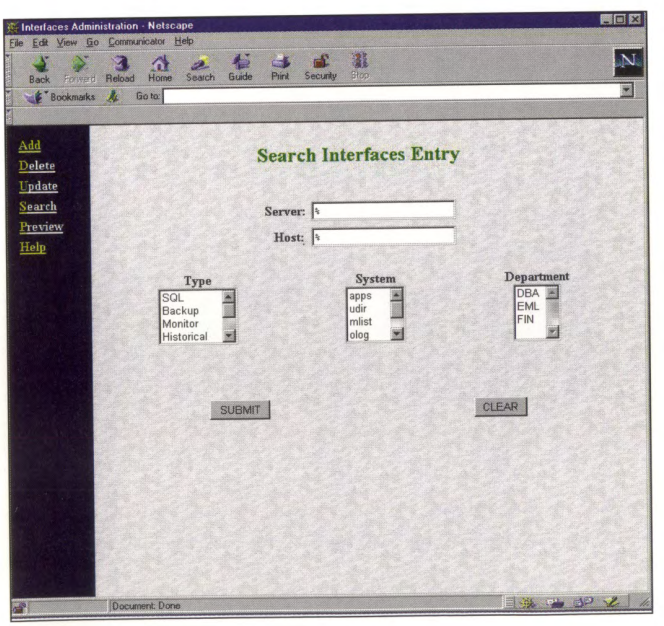


Figure 3. Search Interfaces Entry Web Page

Interfaces File Generation and Distribution

Two Perl scripts, *make_interfaces.pl* and *cvt_interfaces.pl*, are used to generate the interfaces files. The first one reads the database and formats the data into a standard Unix interfaces file. The second Perl script converts the standard interfaces file created by the first script to other formats (Sun Solaris or Windows) for use on different platforms.

Let's take a look at the interfaces file generation script first. It starts by declaring the data structure to identify what interfaces files to create, which server entries to include, and the order in which entries appear in the file. Two representative examples of the database structure are shown below:

```
$sapi_ref = {
file=>"interfaces.gen",
fddi=>1,
types=>["SQL"],
groups=>["apps ", "udir", "mlist",
    "olog", "efin", "generic"],
};

$apps_ref = {
file=>"interfaces.apps",
fddi=>0,
types=>["SQL", "BAK", "MON", "RS", "LTM"],
groups=>["apps", "generic"],
};
```

The above Perl record structure actually defines interfaces file generation rules for each application group. **file** defines the name of the interfaces file to be created. The suffices **.gen** and **.apps** determine the computing environment where the interfaces file will be distributed to. For instance, **interfaces.gen** will be distributed to clients and servers to support the application named **gen. fddi** indicates whether fddi network entries will be included in the interfaces file. (fddi is a subnetwork in our environment. The applications on this network listen to ports on two networks.) **types** indicates the type of servers to be included, which corresponds to **srvr_type** in the database table. The first record structure includes SQL type, indicating that Adaptive Server entries are to be included in the interfaces file named **interfaces.gen**. The second record structure indicates that all server types are to be included in the interfaces file **interfaces.apps. groups** indicates the application groups to be included, which corresponds to *subsystem* in the database table.

When the data is extracted from the database and written into the interfaces file, it is sorted by the *groups* and then *types* fields. The second record structure indicates that all Adaptive Server entries from the *apps* application will be listed before all entries from the *generic* application. The entries in the same application group will be listed alphabetically in the interfaces files. The ordering is repeated for each type, so backup servers (indicated by type BAK) will be listed after. In a large scale client-server computing environment, the ability to order server entries in the interfaces files is important for performance. Since the applications have to traverse the interface file looking for entries, frequently visited server entries must be listed first to minimize the scan time.

The version control and source management capabilities are also added to *make_interfaces.pl* scripts. Multiple versions of application-specific or global interfaces files are maintained by RCS so that modification errors on the files can be remedied by checking the previous and correct versions. It also allows us to track changes to the files. RCS checking in and out functionality is built into the script in order to save system administrators from managing versions manually.

The following is the kernel of the script to generate interfaces files:

```
unlink $file;
print "creating new $file\n";
#create an interfaces file
open(INTERFACES, ">$file");

#write RCS keywords into the file
print INTERFACES "#\n";
$ident =~ s/\s+\$Id/\$Id/;
print INTERFACES "# $ident\n";
print INTERFACES "#\n";

#loop through each type of servers to
#be included in the interfaces file
for $type (@{$subsystem->{types}}) {
 #loop through each application group
 for $group (@{$subsystem->{groups}}) {
  #check fddi network entries
  if ($subsystem->{fddi}) {
   $cmd = <<"EOSQL";
     select server, query, protocol,
         network, hostname, port
     from interfaces
     where subsystem = '$group'
      and srvr_type = '$type'
     order by server, query, rownum
EOSQL
```

```
    }
    else{
     $cmd = <<"EOSQL";
       select server, query, protocol,
            network, hostname, port
       from interfaces
       where subsystem = '$group'
        and srvr_type = '$type'
        and hostname not like '%fddi%'
       order by server, query, rownum
EOSQL
    }

    #execute the SQL command
    $server_data = $dbh->sql($cmd);
    #read each row
    for $server_info (@$server_data) {
     ($server, $query, $protocol, $network,
      $hostname, $port) = @$server_info;
     #server name to be listed once
     #in the interfaces file
     unless($srvrs_done{$server}){
      print INTERFACES "$server\n";
      $srvrs_done{$server} = 1;
     }
     print INTERFACES "\t$query $protocol $network $hostname
 $port\n";
    }
    #clean the command
    undef $cmd;
   }
  }

  close(INTERFACES);  #close the interfaces file
```

After the interfaces file is created, it is checked back into RCS. The script generates all application-specific and global interfaces files. The *rcsdiff* utility included in the scripts detects whether an interfaces file has been changed compared to the last build. If there is no change, the new interfaces file is not checked into RCS and any locks on the file are released.

The second script, *cvt_interfaces.pl*, converts standard Unix interfaces files to Sun Solaris or Windows format. The following Perl procedure takes a standard socket format interfaces file entry as its input and converts it to Sun Solaris format:

```
#input arguments.
my($query, $hostname, $port) = @_;

#get the host IP address.
($he_nm, $he_al, $he_ty, $he_len, $he_ad) =
  gethostbyname($hostname);
@ip_dec = unpack('C4', $he_ad[0]);

#convert the IP address to HEX,
#padding with zero at front
$ip_hex = sprintf("%02x%02x%02x%02x",
        $ip_dec[0], $ip_dec[1],
        $ip_dec[2], $ip_dec[3]);

#convert the port number to HEX,
#padding with zero at front.
$port_hex = sprintf("%04x", $port);

#print out the TLI entry
print "\t$query tli tcp /dev/tcp
\\x0002${port_hex}${ip_hex}0000000000000000\n";
```

The following Perl procedure takes a standard Unix format interfaces file entry as its input and converts it to Windows format:

```
#input arguments.
my($query, $hostname, $port) = @_;

#print out the INI entry
print "${query}=NLWNSCK,$hostname,$port\n";
```

The above scripts show how simple it is to retrieve data stored in a database and convert the data into any interfaces file format. Future directions will incorporate both scripts into one, having all interfaces files generated in all formats.

The Push Method

After the interfaces files are created, the next step is to distribute them to the computing environment. Application-specific interfaces files are distributed to clients and servers supporting the application and the global interfaces file is distributed to certain hosts where connectivity to all servers are necessary. Correct formats are copied to hosts according to their platforms. Since all interfaces files are distributed from a central location, it is called the "Push method," which essentially

uses the RCP or FTP utility on UNIX platform. In the meantime, when interfaces files are generated from data stored in the database, a series of push scripts are created based on column *hostname* and *subsystem* in the table. These scripts are responsible for copying correct interfaces files to corresponding hosts.

The push paradigm benefits from its ease of implementation. All scripts are maintained at a central location. All created interfaces files can be verified before being distributed to the environment. When there is any problem with the interfaces files in the environment, system administrators can modify the entries stored in the database via web, then generate and distribute the correct files to resolve the issues right away.

The Pull Method

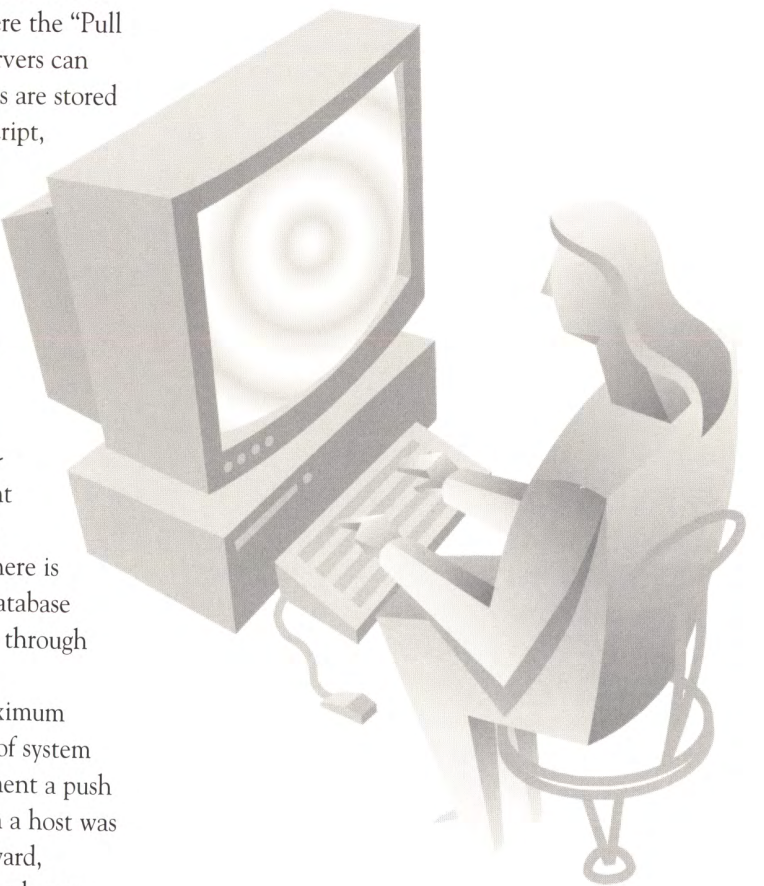
Though it is straightforward to implement the push method, we encountered problems pushing interfaces files to hosts residing outside the corporate firewall (mainly web servers). In order to maintain tight security, these hosts normally don't allow RCP or FTP connections. However, they can connect to database servers inside the firewall. That is where the "Pull method" comes into play. Essentially, clients or servers can directly query the database where interfaces entries are stored and generate the interfaces files on the hosts. A script, similar to *make_interfaces.pl*, makes this possible. System administrators can schedule these scripts to execute in order to populate the changes from the database to interfaces files on the hosts.

Since the pull method relies on an individual script on each host, it eliminates some limitations of the push method. Programming logic can be built into scripts on certain hosts to create cross-application-boundary or other special type of interfaces files. However, the scripts residing on different application hosts must also be different in order to create application-specific interfaces files. When there is a need to modify these scripts (such as interfaces database or server name change), it is not an easy task to go through each host making modifications.

From the outset, our design goal was to put maximum control of distributing interfaces files in the hands of system administrators. Because of this, we chose to implement a push strategy so that any updates to the interfaces file on a host was decided on by the system administrator. Going forward, combining the push and pull paradigms looks to be a better solution in our environment. System administrators can still create, verify, and distribute the interfaces files to most of the hosts using push method. A limited number of hosts can have the liberty to create their special interfaces files using pull method. The combination creates a reliable interfaces file administration mechanism in a secure environment.

Conclusion

We have been using the web and backend scripts to maintain our interfaces files for over a year. During this period, we have virtually eliminated errors to interfaces files, where the errors were from contents or formats. This significantly improves availability, robustness, and performance of our overall systems. Our group works in a very fast-paced environment, where the network is reconfigured on an ongoing basis to accommodate system expansion. Our web-based interfaces files administration mechanism allows us to adapt quickly to these changes. Using the web as an administration tool has been a boon to administering interfaces files, and will continue to provide better productivity as new tools continue to be rolled out. ■



Global Peer-to-Peer Data Replication Using Sybase Replication Server

By Mich Talebzadeh

I recently worked on the implementation of a global peer-to-peer data replication for a Front Office Trading System (FOTS) utilizing Sybase replication. This work was carried out as part of a major requirement for a global investment bank. The FOTS provides traders with a view of the trading activity and the positions held, allowing them to continue trading against orders placed earlier by other offices in other locations. For example, a trade may include the exchange of securities in multiple geographic regions such as London, New York, Tokyo, and Hong Kong.

The work was actually started by building a Sybase Warm Standby for the London database. Having successfully implemented this utility, a pilot project was put in place to check the feasibility of one-way replication to the Hong Kong Data Server, effectively testing the volume of data replicated and stress-testing the WAN. As the application relied on a third-party package, care was taken to avoid changing the code. The data structure was enhanced by addition of primary keys (prerequisites for warm standby replication). In peer-to-peer replication, a given database acts as publisher and subscriber simultaneously; all databases play equally important roles.

Having identified the initial problems, a truly peer-to-peer replication system was set up among London, Hong Kong and Tokyo, effectively replicating almost all the tables. The information on all sites had to be as current as possible and had to be available 24 hours a day. Practically in excess of 95% of the transactions had to be applied to all databases, worldwide, within five minutes of data being posted to a local database.

This article provides templates of how to create replication and subscription definitions for user tables in a given Sybase database, as well as practical examples of how to apply function strings to tailor what is delivered to the destination table. In addition, we'll discuss ways of monitoring the replication system and quickly checking the data.

Project Problems to be Addressed

A typical trading system needs to provide facilities to a business community scattered around geographically distant sites; in our case, London, Tokyo, Hong Kong and New York. A trader logs into the Application Server (a UNIX server) locally through an X-Windows session and starts the application, which connects to the Data Server via one or more threads. In the majority of cases this set-up involves many Application Servers running locally, with the Data Server located in one of the major sites. In our example, the Data Server was located in London. However, this set-up has the following drawbacks:

- Application performance is limited because of the geographical distance.
- Network response degrades when traffic over the WAN is heavy. For exam-

Mich Talebzadeh is an independent Sybase consultant, working largely in investment banking. He teaches on topics including SQL Server administration, performance and tuning, data replication, and client/server database and application design. He can be reached at: mitch.talebzadeh@db.com

ple, when users in remote sites run reports requesting a large number of rows, there is an impact on those entering trades etc.
- The Data Server can potentially become a bottleneck as a larger user community contends for access.
- Data becomes unavailable when a failure occurs on the network.
- There are side effects on database maintenance. For example, Update Statistics, Database Consistency Checks (DBCC), and other tasks are performed when remote users are busy putting in trades. This introduces unnecessary complications and delays.
- A typical third-party application may not access data in the most efficient way. For example, if the application makes a large number of discreet queries to the database, connection latency between the Application and Data Server could cause start-up delays.
- If the application were to crash, the system becomes unusable to traders and has an unacceptable business impact.

We responded to these issues with a three-step plan. First, we created the Sybase Warm Standby database for the local site on the standby (BCP) server. Next, we created a peer-to-peer replication system for the trading database with local copies of the database in London, Tokyo and Hong Kong respectively. The database in each site acts as the source and recipient of data—in other words, the database plays the role of publisher and subscriber simultaneously. Third, we created a set of utilities to monitor and maintain the replication system. In such a trading system, service availability and recovery is essential, so particular attention was given to early warning systems.

Where to Locate the Replication Server

The Sybase Replication Server is an open server and could be located on any UNIX Server. It is advisable *not* to put the Replication Server on the production Data Server. This keeps the production box simple and less complicated (a plus for maintenance and recovery), allows us to resource the CPUs and memory for the production SQL Server, and keeps the Replication Server up and running even when the production box is down. The Replication Server was created on the BCP server, which served largely as an emergency backup.

The Warm Standby Setup

I'd like to say a few words on our warm standby set-up (valid for v. 11.0.3). A warm standby set-up is a pair of databases, one of which is a backup copy of the other. Client applications update the active database; Replication Server maintains the standby database. If the active database fails, switching to the standby database allows client applications to resume work with little interruption. A warm standby database is basically a simplified form of one way replication.

Sybase Warm Standby replication will only replicate the data to the standby box, and the standby database can only be used in read only mode. Of course, there can only be one warm standby set-up for a given database. Any changes to the objects in the database (i.e., database related application patches) will not be replicated and will have to be applied to the production and standby databases. Login names, database users, and permissions are not replicated. Although Replication Server does not usually require replication definitions in order to maintain a standby database, it does use any replication definitions that exist. Note that you need to generate replication definitions for tables with more than 128 columns.

We also created primary keys for replicated tables. In a warm standby set-up, the Replication Server generates a **where** clause to specify target rows for updates and deletes. If there is no replication definition for a table, the generated clause includes all columns in the table, except text, image, timestamp, and sensitivity columns as the primary key. This turns out to be inefficient.

Also, don't forget that you still need to prune the transaction log of the primary database on a regular basis. Just one option would be to dump the production database daily at 6:00 a.m., followed by dumping of the transaction log at regular intervals between 7:00 a.m. until 9:00 p.m. After 9:00 p.m., turn on the **truncate log on chkpt** option on the production database. You need to fit this to your schedule.

Planning for the Peer-to-Peer Replication System

In planning a global replication system, I suggest that administrators take the following ideas under consideration. First and foremost, of course, some business rules have to be defined in conjunction with the business in order to guarantee the integrity of data on each site.

Then, when beginning to test your system, set up a one-way replication between two locations, identifying the primary site (in our case, London) and the subscriber site (Hong Kong). Of course, the best option is to set up the test environment on one local and one remote server. Otherwise, you can use two different servers locally linked via a WAN simulator. Note that a one-way set-up can be extended to a peer-to-peer replication.

It is important to estimate the volume of data to be replicated daily, as well as the daily growth of your database. Also, you should establish the bandwidth between the two replicate sites, letting you know whether you should embark on replication or if the bandwidth needs to be improved. In addition, you must establish how the replication is going to be achieved. Do you need to:

- Replicate tables and turn off triggers for replicated transactions?
- Not replicate tables and let triggers insert the records?
- Replicate stored procedures?

Does your application depend on timestamps for certain transactions? Remote locations mean different time zones—a data insert at 9:00 a.m. in London corresponds to a replicated transaction of 9:00 a.m. in Hong Kong (this assumes no latency), whereas the local time would be 5:00 p.m. Would this be considered a valid business transaction? If not, consideration should be given to the use of function strings to allow for the local timestamps.

Using Table Replication

On our project, it was decided after testing to replicate tables and turn off triggers for replicated transactions. With this solution we need to identify those tables to be replicated and establish whether they meet the criteria—are primary keys defined, etc.—and we need to exclude local or static tables, which can be loaded at the start-up when the primary database is loaded onto the remote sites. We exclude the identity column from replication, as the remote server will automatically generate identity columns for replicated transactions.

Managing Conflicts in Peer-to-Peer Replication

Managing Inserts

A typical local table will include inserts from local applications in addition to inserts delivered via replicated transactions. In a peer-to-peer set-up, both tables are bi-directional. Designers tend to use unique IDs to identify records in a table, and primary keys or unique clustered indexes are usually built on the unique ID. The unique ID for a given table tends to be a monolithically incrementing 32-bit integer stored in and retrieved from a table, the so-called *table_next_row_id*. Intersite conflicts occur when rows are inserted in a local table and distributed to the remote table. If the remote table already has a record with the same unique ID, the replicated insert will be rejected, and the data at the remote table will be inconsistent with the local table. Possible solutions are:

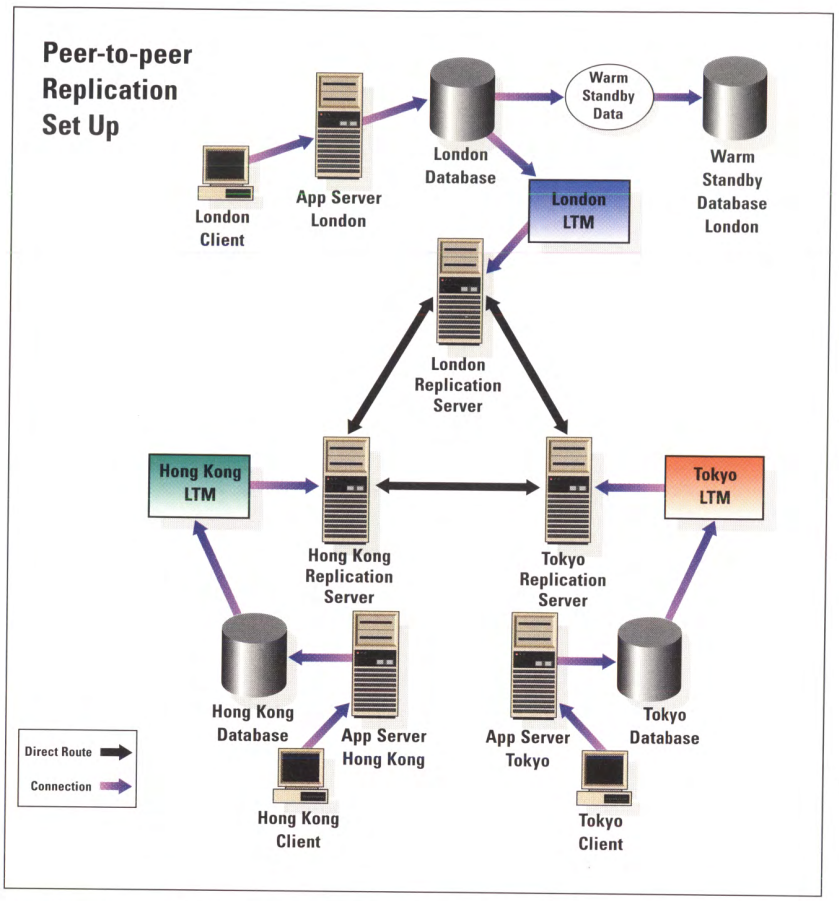
1. Add a location key to the tables if it is not already there, and include it in the primary key for replicated tables. This is useful if the application is at the design stage. As replication implementation is normally an afterthought, this approach may not be possible without a substantial change to database schema and stored procedures (establishing ownership of data).
2. Localize the *table_next_row_id* and do not replicate it. Allocate ranges for the *next_row_id* column for each location (a 32-bit integer provides ability to store up to 2 billion unique values). For example, you can allocate the following ranges:

Location	Reserved range for next_row_id Column
London	1-100,000,000
Hong Kong	101,000,000-150,000,000
Tokyo	151,000,000-200,000,000

Conflicting Updates

The best way to handle conflicting updates from different sites is to construct the application and the environment so that conflicts cannot happen. However, in many cases one needs to rely on application-specific business rules to reduce/eliminate the conflicts. For a trading system these could be:

- On *performing a simultaneous new trade* on the same holding on a different site, the problem will occur on a calculated field, such as quantity or P&L. Furthermore, there are no signals to warn the users when the problem occurs. The adopted solution is to recalculate these fields nightly so that they will have the correct figure by the following day, when the portfolio is loaded. However, we have not yet encountered such a problem.
- On *simultaneous update on the same order*. This could happen due to a mistake. The business rule is whoever created the order owns it, and should be the one who updates the order. If this happens, the quantity (P&L) data will be out of sync. Again, there will be no warning message to indicate this and it will be very difficult for IT support to detect it. The traders will inform Application Support that the P&L or quantity is wrong. The support group needs to check the historical order and amend it appropriately. Once this is carried out, the correct details will be replicated to other sites and the databases will be in sync again.



Building three replication servers using sybase rs_init facility

Peer-to-Peer Replication Implementation

In our set-up, we are replicating data between London, Hong Kong, and Tokyo. The majority of the tasks mentioned below are best performed when no user is using the databases, such as over weekends. Each database is controlled by its local replication server.

Replication Server	Location	Location of RSSD	ID server
lon_rep_server	London, on Standby Server	Standby sql server	yes
hk_rep_server	Hong Kong	Hong Kong sql server	-
tyo_rep_server	Tokyo	Tokyo sql server	-

1. Before creating replication servers ensure that you have already created devices for the RSSD databases and have raw partition devices for the stable queues, etc. In other words, fill in those Replication worksheets.
2. Once the Replication Server is created, change its password using the **alter user** command from the Replication Server.
3. Create the diagnostic run files for replication servers so the DBA may observe each replicated transaction performed by the server (invaluable in identifying problems). This is achieved by replacing repserver binary with repserver.diag binary in the run file and adding the following entry to the replication server .cfg file:

```
trace=DSI,DSI_CMD_DUMP
```

4. Create error class rs_sqlserver_error_class (default sql server error class) in ID replication server only. This will handle sql server errors in replication server. The default error action for all errors returned by sql server is to stop replication. You can assign action to the created error class etc. Error actions are stored in table rs_erroractions. Use rs_helperror error_no, v to get information about the errors.
5. Turn off trigger settings for London, Hong Kong and Tokyo servers in the corresponding replication server. Use configure connection command with dsi keep triggers option set to "off." For example, in lon_rep_server run the following command:

```
configure connection to london_sql_server.db
set dsi_keep_triggers to 'off'
```

6. Add the server name and port ID of replication servers to the relevant interfaces files.

Creating Direct Routes

In our triangle diagram, we need to create routes in order for our three replication servers to send messages to destination replication servers. A route is a one-way message stream that sends requests from one Replication Server to another, carrying data modification commands, replicated functions, and stored procedures. In this design the routes are created as follows: For example in lon_rep_server run the following command to create route to hk_rep_server:

Route type	Source	destination
Direct	lon_rep_server	hk_rep_server
Direct	lon_rep_server	tyo_rep_server
Direct	hk_rep_server	lon_rep_server
Direct	hk_rep_server	tyo_rep_server
Direct	tyo_rep_server	lon_rep_server
Direct	tyo_rep_server	hk_rep_server

```
create route to hk_rep_server
set username hk_rep_server_rsi
set password hk_rep_server_rsi_ps
```

hk_rep_server_rsi is the RSI username already created by rs_init when you created the hk_rep_server. *hk_rep_server_rsi_ps* is the default password for such user etc. Use *rs_helproute* in any RSSD to check the status of the route created.

Loading the Database to Be Replicated

In order to perform the initial load of the database to be replicated, you should perform the following steps:

1. Decide where you are going to load your initial database. In our case, we chose London.
2. dbcc the database and perform update statistics in London.
3. Review all the primary keys for tables to be replicated.
4. Turn off all replication flags in the user tables using *sp_setreptable table_name, false*.
5. Do **dbcc settrunc(ltm,ignore)** on the database.
6. Dump transaction with truncate_only.
7. Dump database to the dump directory.
8. FTP the dump file to the warm standby server.
9. Load the database on the standby.
10. Zip the dump file to remote locations.
11. Load the database from the dump file in remote locations.
12. Localize the so-called local tables. For example, if you have *table_next_row_id, set next_row_id* column to the appropriate starting values for location.

Adding databases is quite straightforward: To add to the London Production database, we use *lon_rep_server*; for Hong Kong, use *hk_rep_server*; and for Tokyo, use *tyo_rep_server*. Note that all three databases are a source of data and therefore require an LTM or a rep agent.

Creating Replication Definitions

Once the databases are loaded, we can create all the replication definitions for the London database tables using lon_rep_server. You need to pass the *sql server name* and the *database name* as parameters. For a script (genrepdef.ksh)that will automatically create replication definitions, see the extended version of this article on the ISUG website at www.isug.com. Also, check that replication definitions are successfully implemented by looking at the log files and using rs_helprep in the relevant RSSD database.

To create replication definitions in the other publisher sites, this would be:

Replication Definition	Sql server named passed to script	Database name passed to script	Replication server run against
London tables	london_sql_server	db	lon_rep_server
Hong Kong tables	hk_sql_server	db	hk_rep_server
Tokyo tables	tokyo_sql server	db	tyo_rep_server

Next, turn on replication for all replicated tables by running *sp_setreptable table_name, true*. If you use the command *rs_helprep* in any RSSD database, you should see all replication definitions for all sites irrespective of which RSSD you are looking at. Finally, create subscription definitions for all other sites by running the above script (*genrepdef.ksh)* against the local replication server.

Creating Subscription Definitions

You should create two subscription definitions for each replication definition:

Subscription Definition	Sql server named passed to script	Database name passed to script	Replication server run against
London to Hong Kong	hk sql server	db	hk_rep_server
London to Tokyo	Tokyo sql server	db	tyo_rep_server
Hong Kong to London	london sql server	db	lon_rep_server
Hong Kong to Tokyo	tokyo sql server	db	tyo_rep_server
Tokyo to London	london sql server	db	lon_rep_server
Tokyo to Hong Kong	kh sql server	db	hk_rep_server

The three stages of subscription include creation, activation, and validation. Be sure to check the status of subscription following each stage:

1. The script *gensubdef.ksh* referred to above will automatically generate the subscription definitions for London database tables in Hong Kong. This script can be easily amended to create subscription definitions for any site.
2. Once the subscriptions have been defined, check their status by running *rs_helpsub* in the RSSD database for the rep server. This should show the status as defined.
3. Create and run a script called *activatesubdef.ksh* based upon *gensubdef.ksh*. Replace **DEFINE SUBSCRIPTION ${dbtable}_${DATABASE}_1** with **ACTIVATE SUBSCRIPTION ${dbtable}_${DATABASE}_1**
4. Run rs_helpsub again. This should show the status as active.
5. Create and run a script called *validatesubdef.ksh* based upon *gensubdef.ksh*. Replace **DEFINE SUBSCRIPTION ${dbtable}_${DATABASE}_1** with **VALIDATE SUBSCRIPTION ${dbtable}_${DATABASE}_1**

6. Run rs_helpsub again. This should show the status as valid.
7. Repeat the subscription definitions for other sites as well.
8. At the end of subscription definitions, you should have two subscription definitions for each replication definition. In other words, doing rs_helpsub for each table should give you 3x2 subscriptions = 6 lines. This should be shown in any RSSD database.

Use of Function Strings to Apply Local Timestamps

Replication Server converts functions to commands and submits them to destination data servers. For example, a new row inserted in the source table causes Replication Server to distribute an *rs_insert* function specific to that table to the subscriber databases. A possible solution for applying local timestamps at replicate database would be to modify *rs_insert* for a given source table to invoke an RPC at the destination database. The RPC in turn inserts local timestamp to the subscribed table. Note that for a peer-to-peer Replication System this process needs to be applied to all databases.

Handling Replication Maintenance

The rs_init facility automatically creates a maintenance login in the format of *database_name_maint*. This user is created as a public user in the replicate database. If you are using a warm standby set-up and your tables contain identity columns, you need to make user *database_name_maint* a "dbo" in the standby database, as this user needs to set the option *identity insert on* when replicating tables with identity columns.

Tuning Replication Servers for Better Performance

There are some configuration parameters that can be altered in order to get better performance from the Replication Servers:
init_sqm_write_delay: A stable queue manager waits for at least init_sqm_write_delay milliseconds for a block to fill before it writes the block to the correct queue on the stable device (default is 1000). Try decreasing this parameter.
init_sqm_max_write_delay: A flush to the queue is guaranteed to happen after waiting for init_sqm_max_write_delay, if the DSI or RSI thread reading the queue is unable to connect to the target or has been suspended (default 10000). Decrease this parameter if required. *sqt_max_cache_size:* You will need to increase this value if there are a lot of open transactions and or large transactions. Memory for sqt_max_cache_size is taken from the global memory pool (default is 131072 bytes).
batch_sz: The larger the batch_sz, the less often the truncation point is updated (default 1,000 commands).
This parameter is set in LTM configuration (cfg) file. Applicable to Replication Server up to 11.0.3.

Monitoring the Replication System

Replication Server provides a host of commands for checking replication status:

- Replication Server commands
- Replication Server Manager
- Sybase Central (Replication Server 11.5.1 and above)
- rs_subcmp program that allows comparison of tables in two replicate databases and has flags for reconciling them.
- Specifically written scripts

Beware of the use of *rs_subcmp* for reconciling large tables between remote databases. This may take a long time and will not always be practical. You may consider BCP'ing data instead. Also see the code posted with this article at www.isug.com for additional ideas.

Monitoring the Latency and Delivery of Data

Finally, it is a common practice for DBAs to set up a table in the replicated database where data is updated for the purpose of checking the latency and health of the replication system. In its simplest form, one can insert or update records in this table and see if the data is being replicated to the other sites. The time it takes for data to get to the remote site will give an indication of latency. However, it doesn't indicate whether the business transactions arrive in remote sites in a timely manner; nor does it allow for applications creating a large number of transactions where data delivery is impacted by concurrency, table size, or any locking mechanisms. It should also be noted that the maintenance user trying to deliver the replicated data may be blocked by local users. If the statistics on the user tables are not current, the replicated data may take a longer time to be delivered, resulting in remote users being blocked waiting for locks to be released.

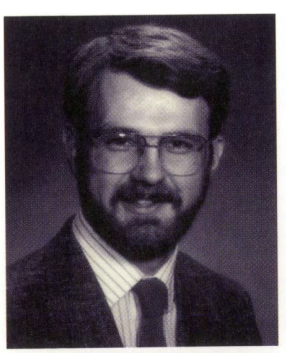
Therefore, it is important to have a more realistic method for evaluating replication delivery. A possible solution is to look at the entries in an audit or transact history table in the replicate database and check the delivery timestamp. Any latency can be estimated by comparing the timestamps for records delivered and adjusting for server clock differences.

Syntax referred to in this article is available for download in the Members Only section of the ISUG website at www.isug.com. ■

Programming Java Inside Sybase's Adaptive Server Databases

By Sam Blanchard

This is the second half of an article exploring the question: Is Sybase's Java implementation a positive addition to the Adaptive Server family?



Sam Blanchard is a senior relational DBA at Ball State University, where he has worked for the past four years. He is currently finishing a master's degree in information and communication sciences at Ball State. He can be reached at sam@bsu.edu.

The Adaptive Server family of databases now incorporates Java within the database server. With all the hype about Java, it is important to evaluate the potential impact of Java in these databases. Java in the database is positive if it provides advanced programming features without adversely affecting core database objectives. But it will have a negative impact if it does not integrate well, increases complexity beyond any benefits or adversely affects core database objectives.

The first half of this article (which appeared in the 1st quarter 1999 issue of the *ISUG Technical Journal*) explored the concepts of data independence vs. object independence and how these relate to achieving core database objectives, provided an overview of new features, and went into some detail on programming Java within the database. My conclusion is that a good Java solution must be focused toward data independence, while allowing the DBA to utilize object independence as needed.

This second half of the article moves into development issues relevant to both programmers and DBAs. As with any combination of technologies, it may be necessary to get both DBA and Java programming skill sets together to take full advantage of this article. But even if you aren't an expert in either technology, the material from the previous issue should suffice to make this both a starting point and future resource.

The Java Development Environment

Sybase's Java in the database does not, in fact, provide a Java development environment, though it does allow debugging. The Java Virtual Machine (Java VM) deployment environment supports code developed to Sun Microsystems's Java Development Kit version 1.1 (JDK 1.1). However, Sybase does offer PowerJ as an individual, visual Java integrated development environment (IDE), or, if you are already developing Java, start by using your current tool.

Unless you already know Java well, consider working with a Java developer who can create the initial Java classes. For our initial development, we used JDK 1.1.7. If you are not currently developing Java yourself, you may download the current version for free from http://java.sun.com. The command line interface and your favorite text editor will get you started. I did do some JDBC connectivity testing with PowerJ 3.0 and found it very useful for visual development.

What is JDK?

The JDK is actually two things: the specification and a development tool for Java. Sybase created its Java VM to run a subset of JDK 1.1.6, which is generally compatible with any JDK 1.1.x version. The newest JDK is 1.2 (called Java2) but support for this version is not available in Adaptive Server Anywhere 6.0 at this time. Sybase is leaning toward Sun's

Java2 Embedded Server as a possible direction. Reviewing the Java2 Embedded Server home page (see http://www.sun.com/software/embeddedserver) will provide you with an understanding of how Sybase envisions Java in the database. There will naturally be some tradeoffs between supporting every possible Java extension on the one hand, and, on the other, providing a small footprint solution that can scale down to Personal Digital Assistants (PDAs) and other applications. (It is inevitable that Java will fragment slightly on issues like embedded systems vs. large scale applications. The functionality may vary slightly, but the implemented functionality must maintain compatibility to a single, overall standard.)

What is a Java Object?

Java objects are coded as class definitions, including the fields (actual data elements) as well as the methods (functions and subroutines). There can be several layers of definitions depending on how classes contain, extend, or modify other classes. Suffice it to say that Java objects can be quite complex. I only use simple examples of Java objects here, so don't be surprised if a Java programmer comments on the simplicity of these examples.

In object-oriented terminology, the allocation of space associated with a class is called *instantiation*. When a Java class is instantiated, the environment allocates space in memory to hold a new Java object consisting of the fields and methods. Each time space is allocated for a Java class, another Java object is created. Each instantiation creates a unique object that can change over time.

Java Classes and Sybase User Defined Types

A Java class acts as a new type within Adaptive Server. From a data management perspective, it's easy to see the field values of an object as independent of the class definition. A pure object-oriented approach does not separate the fields from the class definition. Java, being object-oriented, explicitly keeps the data with the methods; however, relational data management presents a conceptual problem when fields and methods are stored together. This is one area where object-oriented databases and relational databases embrace different philosophies (column-level data independence vs. object-level data independence). Java serialization provides specific rules for the proper storage and retrieval of an object, and Adaptive Server follows Sun's JDK serialization rules with some modifications. Adaptive Server does keep the class definition and the data together in the database when using objects in the database.

Java Field Types and Sybase Column Types

Each Java field in a class has a type, just as Sybase has columns and variables. Java's primitive types are similar to Sybase primitive types, and Sybase provides a compatibility and conversion table. The Java primitive types are defined by the JDK and implemented by Sybase; Sybase cannot change the primitive types and still provide a compatible environment.

Like Sybase user-defined types, Java allows new field types to be created. Note that any Java class loaded into the Sybase Java VM automatically becomes a user-defined type in the Sybase database server. Each environment provides sufficient flexibility to support the other environment's data types.

Java Methods Within the Database

A Java method is an executable section of code that exists within an object. Within the database server, a method is available as a Sybase user-defined function. The ability to easily add functionality within the database server via Java provides a lot of potential. However, additional functionality in the database server also adds complexity. Prudence dictates that database administrators carefully consider all the ramifications of this.

Java methods run like user defined functions in the Java VM within Adaptive Server. Database performance and maintainability may be severely impacted by unwisely coding or invoking Java methods; the same problems can occur with traditional stored procedures. Sybase/Java interoperability also provides the ability to program outside the database and move critical components into the database for specific reasons. Our own current work provides small functional enhancements to the Sybase ASA Server. In specific cases, it may be appropriate to code more of an application in the database. Be sure that your deployment scheme includes the ability to migrate code into and out of the database server as appropriate.

SQL/J—Its Future with Sybase

SQL/J is a joint effort between several database vendors (Oracle, Sybase, IBM and others) and Sun to provide a standard embedded Java for portability and power. Since SQL/J is not a completed specification, you won't see Sybase technical staff in Canada or California calling anything SQL/J just yet. I dug deeper to see both what SQL/J is and what real applications are in the works at Sybase.

SQL/J includes specifications for embedding SQL statements in Java programs, for installing Java classes in a database and calling the class methods, and for supporting Java classes as user-defined types following existing SQL standards.

The Watcom-SQL extensions we worked with in the first half of this article are an implementation of the current SQL/J syntax. Sybase developers are very proud of their class-based implementation in Watcom-SQL (rather than a pre-compiler based solution). The class-based solution takes this SQL/J implementation beyond compiled embedding of SQL in Java or of Java in stored procedures. Sybase should continue using the power of Java classes in the database to provide SQL/J features. Also expect Sybase to provide the option of pre-compiler-based SQL/J.

The PowerJ folks are also considering support for SQL/J development via PowerJ. Look for easier debugging of Java inside the database from PowerJ. Based on my experience of just typing the code for this article, I'm definitely hoping for a greater interactive capability to support object management and debugging. PowerJ should offer the traditional Sybase commitment of interoperability, along with special features to feed into the Adaptive Server databases. PowerJ development and deployment of Java classes in the database would be required to keep Adaptive Server competitive.

SQL/J will probably pop up in other places where you might see embedded SQL. I suspect someone will have Object-COBOL with embedded SQL via an SQL/J-like syntax. Expect most vendors to provide only the pre-compiler method. For SQL/J, calls are generally converted to JDBC calls prior to static compilation. The Watcom-SQL Java extensions provide an efficient class-based implementation, allowing you to enter the SQL/J commands interactively. Expect both Sybase stored procedures and PowerJ Java programs using SQL/J calls to access Java features efficiently.

What is JDBC?

Java Database Connectivity (JDBC) is a special Java class that implements a standard SQL database-access interface, allowing Java programs to communicate with a relational database. JDBC provides a consistent interface similar to the ODBC interface used by many other development tools. Note that any data management system can implement a JDBC interface.

Let's extend the class Example.Widget from the previous article to include two new methods to access data via JDBC. We'll just show part of the class here. Each method will establish a JDBC connection within the database, issue an SQL call, and use the query result to set the field values. This time, I've imported the java.sql package that contains the JDBC API. I've also implemented serialization.

```
import java.sql.*;

public class Widget implements java.io.Serializable
{
 // variables or properties

 private int                uniqueStorageValue = 0;
 private int                companyID          = 85;
 private String  widgetName        = "Default Widget Name";
```

The method loadWidgetOne selects from the traditional relational table WIDGET_ONE and sets the object fields. I've added comments here to identify the different URL options, and highlighted the select statement and the field assignments. JDBC provides a number of getXXX() methods to access query results.

```
public void loadWidgetOne(int loadCompanyID){
  Connection con = null;
  Statement select = null;
  ResultSet result = null;
  con = DriverManager.getConnection("jdbc:default:connection");
    // with 6.0.0 use the empty string ""
    // with jConnect use "jdbc:sybase:Tds:localhost:2638"
  select = con.createStatement();
  result = select.executeQuery(
   "select WIDGET_ID, COMPANY_ID, NAME "
   +" from WIDGET_ONE "
   +"where WIDGET_ID = "+ loadCompanyID);
  result.next();

  uniqueStorageValue = result.getInt("WIDGET_ID");
  companyID = result.getInt("COMPANY_ID");
  widgetName = result.getString("NAME");
  return;
 }
```

The method loadWidgetTwo selects from the object holding relational table WIDGET_TWO and sets the object fields. I've highlighted the select statement and the field assignments. Note that the WIDGET_ID must also be selected. WIDGET_TWO is designed so that the system-generated unique identifier is generated by the database. There is no easy way to move the WIDGET_ID into the uniqueStorageValue field when the object is originally

inserted into the table. By centralizing WIDGET_TWO access to loadWidgetTwo, this minor inconvenience is handled in one central location.

```
public void loadWidgetTwo(int loadCompanyID)
 {
  // this method uses the depricated URL
  Connection con = DriverManager.getConnection("");
  Statement select = con.createStatement();
  ResultSet result = select.executeQuery(
   "select WIDGET_ID, WIDGET_OBJECT"
   +" from WIDGET_TWO "
   +"where WIDGET_ID = "+ loadCompanyID);
  result.next();

  Widget resultWidget =
(Widget)result.getObject("WIDGET_OBJECT");
  uniqueStorageValue = result.getInt("WIDGET_ID");

  companyID = resultWidget.getCompanyID();
  widgetName = resultWidget.getWidgetName();
  return;
 }
}
```

Sybase's Java VM in the database server includes the JDBC API. The internal connection uses the special URL: jdbc:default:connection; the 6.0.0 product used an empty string for the URL (this is available now but will not be in the future and should be avoided). Sybase's Java VM resolves JDBC calls and provides results—there is less overhead since the Java VM is a thread within the database address space.

A new feature of ASA 6.0.1 is the ability to connect to other JDBC sources outside of the database. Here's a sample URL: jdbc:sybase:Tds:remotehostname:2638?Service Name=asademo. The default port number, 2638, is reserved for ASA servers. The example assumes that the machine was started as:

```
dbsrv6 –n asademo "c:\asademo.db"
```

Please consider the impact of this feature and only use JDBC access to external sources in a very conservative way. Working in the database address space provides a lot of performance improvement for internal JDBC connections. External connections could have a significantly negative impact on your database performance. There is also probably some complexity in tuning and controlling connections to external JDBC sources from within the database. Expect to see some clarification from Sybase on these issues in the future.

You will need to have the JDBC class(es) in your development environment. JDK 1.x does include the JDBC 1.1 classes in the java.sql package and provides all you need to develop applications to run inside the database server. Sybase offers a JDBC driver, jConnect, which allows Java programs outside the database to connect and communicate with ASA and ASE servers.

Evaluating New Features Against Core Database Objectives

Java in the database does provide the opportunity for a new level of program portability. Code can move between the database and an application server. With Java in the database offerings from Sybase, IBM, Oracle, and other vendors, Java code may be portable between vendor databases. These are positive benefits if these advanced features do not adversely affect core database services.

Java applications can effectively use both object independence and data independence within the database. Stored procedures or application logic are frequently implemented to improve data independence, and Java class definitions can provide equal or superior data independence. Complex objects within the database will provide more problems, but the ability to store objects within the database simplifies the management of less rigidly formatted data. Objects can provide additional data independence where string, text, and blob columns have traditionally been used. Providing limited inquiry and manipulation of specific columns will improve data independence.

Sybase's implementation of Java classes in the database does not adversely alter the database. The option to not load Java into a database is available, as is the option to not make Java available to the server. On the other hand, Sybase's implementation of Java objects in the database does not adequately integrate with the database, and appears to lack support for mapping objects to traditional database tables. It requires administrators to manage redundant data to support relational administration, such as indexing. These problems should be addressed so that traditional table relational operations and administration can be performed on objects. User knowledge of the object will assist in correctly using and administering objects; just as knowledge of a table assists in correctly querying or indexing that table.

Java Beans, Reflection API, and Object Introspection

The Java Bean architecture, the Java reflection API, and object introspection may provide the necessary features and techniques to better integrate Java with the database. Java Beans are Java programs coded using a component architecture. The Java reflection API allows run-time analysis of Java objects by identifying properties and associated methods. Java introspection allows object developers to explicitly specify information about an object, including properties, methods, and associated descriptive information. Further work is needed to determine how to allow objects to maintain object independence while still allowing sufficient access into the objects' internal structure to support intelligent indexing, etc. The final storage format, object or table, should be a data administration task transparent to the developer.

Given Sybase's depth of expertise in middleware and integration issues, the integration of complex Java objects in the database is achievable. Java Bean programming techniques should be followed, the Java reflection API should be integrated, and object introspection supported. These techniques can support an effective integration of object independence and data independence. Consideration should be given to additional object interfaces such as CORBA. Properly designed and implemented, Java in the database should be the best game in town.

Conclusion

Java and Watcom-SQL interact very nicely in the new Sybase environment. Although Java is currently only available in ASA, I look forward to Java in the ASE database this year. Many database professionals will continue as before to work only with SQL and relational data. Others are already working with object programmers and are probably eager to understand some of these new features. Certainly most of us look cautiously forward to more open and feature-rich Java stored procedures. ■

CORRECTION

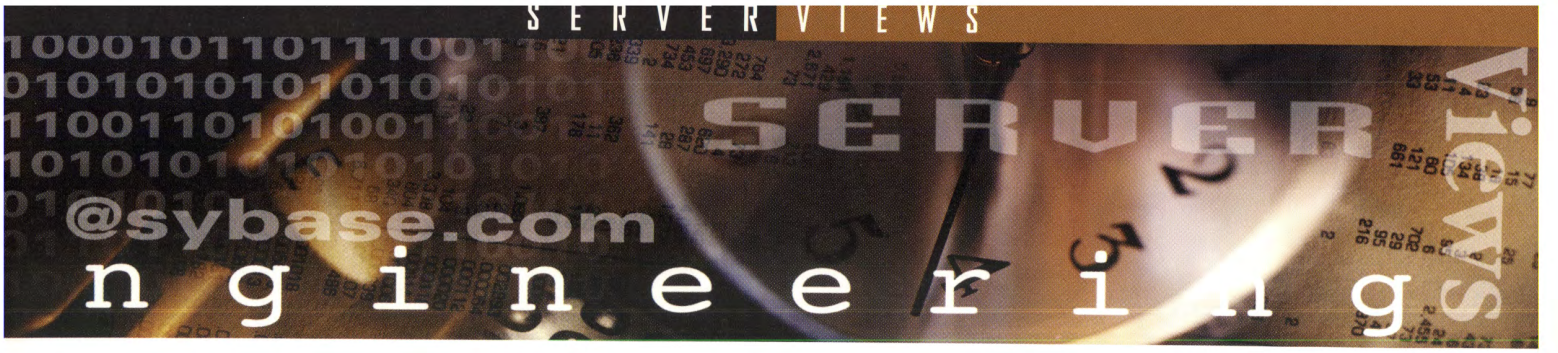
There is a correction to a segment of syntax from the first half of this article. We printed:

```
Create table WIDGET_TWO(
WIDGET_ID int PRIMARY KEY default AUTOINCREMENT,
WIDGET_OBJECT Example.Widget);

INSERT WIDGET_TWO(WIDGET_OBJECT)
VALUES(My_Widget)

SELECT WIDGET_ID,
WIDGET_ID.getWidgetName as NAME,
WIDGET_ID.getCompanyID as COMPANY_ID,
WIDGET_OBJECT
from WIDGET_TWO
```

Since WIDGET_ID is an integer data type and WIDGET_OBJECT is the field with object data type, the select statement should be:

```
SELECT WIDGET_ID,
WIDGET_OBJECT.getWidgetName as NAME,
WIDGET_OBJECT.getCompanyID as COMPANY_ID,
WIDGET_OBJECT
from WIDGET_TWO
```

Our thanks to Asad Ali for catching this error.



Advances in Performance Diagnostics with ASE 12.0

By Peter Thawley

Welcome back! In this issue, I'd like to introduce the project that has consumed much of my time and soul over the last six months: a new performance diagnostics tool for Avatar, the next ASE release targeted for late third quarter. However, before I totally geek out, I'd like to take a moment address the three individuals who sent in questions to the "Server Views" Challenge. Both longtime DBAs, Jimmy Cartledge, with ConAgra Frozen Foods in Omaha Nebraska, and Arun Kamat, with Deutsche Securities Ltd. in Tokyo, wrote to express their frustration at Sybase's "stealth" marketing. Despite the fact that marketing is the butt of many of my jokes, many of us at Sybase share that frustration.

"Stealth" Marketing?

For a long time, Sybase believed in the theory that by taking the moral high ground, customers would respond positively. While perhaps not a widely adopted marketing theory, many, including myself, believe customers want a partner who could proactively steer them through the minefields of implementing complex systems. I still hold this to be an important ideal; however, it is not mutually exclusive with good marketing. Over the course of the next few months, you will see Sybase reveal a radical shift in our marketing message, one that promotes value not only to the technologist, but more importantly, to the business. As I hope my premier column (Q4, 1998) demonstrated, Sybase is investing heavily in R&D to provide you with great technology and application solutions. You'll just have to stay tuned to see Sybase marketing on steroids.

"Small to Medium-Sized" I/O?

The third individual who sent in a question, Steve Chowles, a DBA at MCI WorldCom in London, asks if we will ever increase buffer pool sizes beyond 16Kb to leverage more advanced I/O hardware like EMC's Symmetrix. Well, Steve, you are right in thinking that larger I/O blocksizes is how one achieves the higher transfer rates that today's disks support! In general, the mechanics of magnetic disk storage have two characteristics that place limits on I/O bandwidth.

One is the number of discrete operations a disk can perform per second. For example, a single 7200 rpm disk can typically do about 120 random I/O operations per second. The second characteristic is raw transfer rate. Many disks today boast sustained transfer rates of 12 Mb/sec. However, these can only be achieved through large blocksizes of sequential I/O. For example, with 16 Kb I/O, those 120 I/Os per second would yield a transfer rate of only 1.875 Mb/sec, yet a 64 Kb I/O could yield 7.680 Mb/sec. On the surface, it appears these

larger blocksizes are important for databases to support.

We will be increasing support for larger blocksizes in a future release, but there are several considerations. First of all, the reason we don't support it today is because of our historical space management algorithms. It's important to realize that when a database task issues an I/O call, it is always to retrieve some number of pages of a single object (table or index). Since today we allocate space to an object in logically contiguous 16Kb units called "extents," we currently cannot guarantee that the second half of, say, a 32 Kb I/O would be for the same object.

To support this, we would need to make either our extent or page sizes configurable by object. We feel the "variable size" aspect is critical to deal with the fact that only certain classes of objects will benefit from this. These include tables which are table-scanned frequently or tables with a lot of sequential read/write behavior (audit tables, transaction logs, text and image columns, Java objects, etc.). Making this a hard and fast option at a server-level will only do more harm than good, as both Microsoft and Oracle have found out!

We will most likely make this change at the same time we make page sizes configurable as well. In addition, there are other characteristics (e.g., fragmentation and memory usage) that sometimes negate the benefits of larger I/O, whereas other facilities—such as async pre-fetch—actually perform better. I will keep you posted as we firm up our plans for these enhancements.

Now, let me turn to a related topic and the main discussion of this column, performance diagnostics.

Performance Diagnostics Defined

In my work with customers and partners on performance tuning of applications and systems, I have experienced first-hand the difficulty many of you face in managing large production sites using today's tools. Too often, there is nothing more than intuition and guessing involved. Too often, one has to hope the problem reoccurs just to be able to spot it. There has to be a better way!

Now, before I am accused of derailing the reputation of any of our systems management partners, I don't fault their tools. I actually believe more than half the problem should be attributed to the ASE database kernel, which does not easily provide detailed performance metrics down to the user and statement levels. Often, the tools that could drill down to this level, such as our own SQL Monitor and Monitor Server, were so intrusive that their use was relegated to reactive problem solving! After writing sp_sysmon without having all the right data to be as useful as I needed, I found a partner who understood the interdependence between performance metrics as well as their statistical significance. They also bring some very innovative visualization techniques to simplify and speed identification and diagnosis of problems.

For several years now, Savant Corporation (www.savant-corp.com) has been shipping an excellent tool, albeit for another (inferior :-)) database product. Savant not only brought their leadership in performance diagnostics and visualization, but worked closely with Sybase and several of our key customers to jointly create a tool that has been functionally customized to monitor and diagnose the unique and advanced capabilities of the ASE database kernel.

In order to support the principles promoted by performance diagnostics, Sybase redesigned its monitoring architecture to support constant analysis of production systems. This requires a non-intrusive approach, even when monitoring at the user or application level. Also, to close the loop between development and operational support, Sybase added roughly 150 new metrics, mostly at the object, process, and statement levels, to help provide the correlation between systemic observations and root cause, irrespective of being caused by application or configuration issues.

Performance diagnostics, Savant's unique approach to system management, is substantially different than pure monitoring. Traditional monitoring focuses on simply reporting the instantaneous value of some performance metric, usually with some nice graphical control. For example, the "cache hit rate" on my "Default Data Cache" is currently 65%. However, this ignores the real question: Is 65% a good cache hit rate? I would suggest that it depends!

We all know that at different times of the day, production systems respond to radically different profiles of workload. Therefore, 65% is a good hit rate only if it falls within a range that is considered normal for that period of time. When a user calls and states that the database is slow, a DBA's first impulse is often to dive right into determining what the problem is. However, since an end-user's perception of time has been proven to be the inverse of their patience, the first step should be to validate that a problem actually exists. Is the database slow as a whole, or is just this user's process that is performing poorly? I suspect that most of your time is spent just in answering this question.

Sybase and Savant: A New Diagnostic Tool

The key to providing useful performance diagnostics is, first, to make it simple to see only real problems and then, once confirmed, to quickly drill down to the root cause.

To verify that a real problem actually exists, the screens

for this new tool have been designed with three major criteria in mind. First, the screen must be able to display problems from across the room so that, at a single glance, performance problems can be seen. This is achieved through the muted blue appearance of the screens. If everything in the database is operating at normal levels, there are no indications of problems (i.e., everything looks muted and blue). However, when performance problems appear, their severity is indicated through the graphical control's gradiation of color, from yellow (denoting areas of possible concern) to a bright red (indicating a component of the system is most likely causing performance problems).

Second, the screens logically correlate related data on the appropriate screens, as well as display information using customized controls light-years beyond common business graphics. For example, the server overview screen contains portions relating to database activity, memory allocation, cache hit ratios, network traffic, and the disk I/O subsystem. By displaying all database components on a single screen, the user can quickly isolate the problem reported and identify potential causes. This makes it much easier to see the impact of one component on another.

Third, the graphical controls are designed to correlate the specific functionality with both its current and historical impact on performance. For example, a graphical control that focuses on specific system functionality, such as disk I/O, brings together numerous statistics related to that single database component—in this case, volume of requests vs. latency for the user waiting on those requests.

A unique aspect of these controls is they always display a performance metric in comparison to a historical norm for that metric. All controls calculate an average and a standard deviation in order to define that "normal" range. In order to account for variations in workload, these statistics can be calculated over a specified time interval, within which valid comparisons can be drawn. The trend intervals supported are:

- Past hour of the current day;
- Same hour yesterday;
- Same hour, last 30 days;
- Same hour, same day, last 30 days;
- Last 24 hours.

Advanced Visualization Techniques

To help DBAs answer that first fundamental question of whether the entire database is slow or it's just the one person who is complaining, the notion of a benchmark is provided. This control is meant to provide a quick and dirty assessment of the entire system's general health and performance. The graph (Figure 1) displays the wall clock time required to execute a custom procedure, developed specifically to measure ASE performance. It displays the past hour's worth of collection points and, like all controls, supports historical replay. The dotted black horizontal line represents your average with the two solid horizontal lines represent a standard deviation. The white vertical line at the right represents the current value. This particular example shows us that the current performance of this benchmark is a little more sluggish than average but still within normal range.

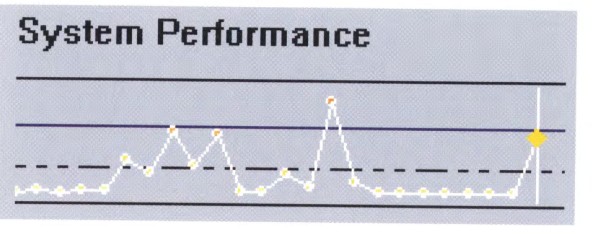


Figure 1

Perhaps the best example of the power of visualization is the hit ratio control (Figure 2), commonly referred to as the eyeball control. The white horizontal axis represents the percentage of data found in cache (0-100). The vertical axis, which gradiates from yellow to red, identifies the current hit ratio. The black X represents the average, and the light circle surrounding the X represents your normal range (standard deviation) for that trend period.

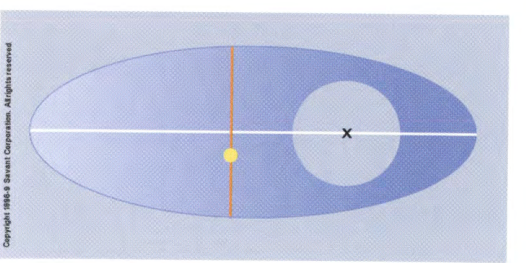


Figure 2

The diameter of the gray circle is significant because the larger it is, the more variance exists in performance of the underlying cache components or in the types of load running against this. Lastly, the colored ball on the vertical axis tells me whether, within the last hour, whether this hit ratio is getting better or worse. If the yellow ball is on the horizontal axis the hit ratio over the past hour has been at average. If this hourly trend is below the average, the ball falls below the

horizontal line, and conversely if the hourly trend is above the average the ball moves above the horizontal line. This particular example tells me my current hit ratio is well below normal, about 40%, and it's been getting worse over the last hour.

When looking at disks, which are a common bottleneck in any system, Savant developed the Queue control, represented by a funnel (Figure 3).

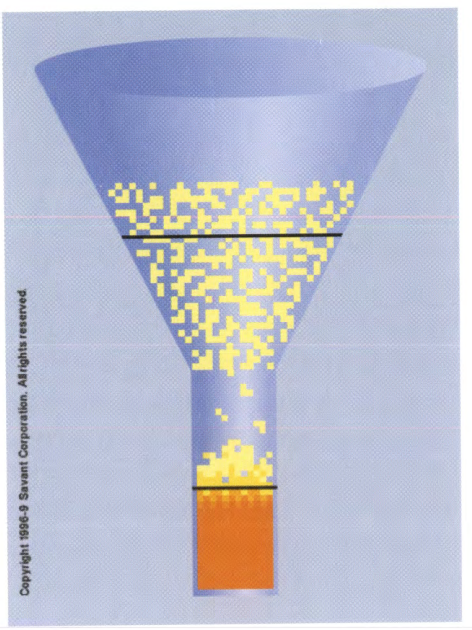


Figure 3

It is used to display information related to processes waiting for I/O requests to be serviced. The top portion of the funnel indicates the number of I/O requests in the queue. The neck of the funnel indicates the average wait time for the requests. The top and bottom portions of the control contain a black line indicating the average for the statistic shown.

The Queue control operates on the principle that volume through a queue does not necessarily have a direct impact on performance. The performance impact of I/O is really derived from its average response time, or latency, while "traveling" through the queue. The most important point this illustrates is that users are waiting longer than normal for I/O, as evidenced by the strong red coloration in the neck of the funnel. In this example, this is most likely due to increased volume of I/O requests, which seems much higher than normal.

Although there are several other innovative controls in this tool, I unfortunately do not have the space to cover them all in this column. I did want to cover one more because of its importance to distributed and replicated environments, the network I/O "tube" (Figure 4).

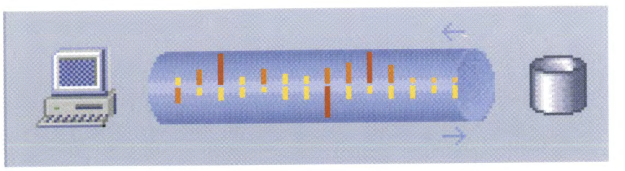


Figure 4

The network tube displays all network traffic flowing both in and out of an ASE Server. The arrows above and below the right hand side of the control shows the direction of flow. Since traffic arriving at the server (i.e., requests) is indicative of different concerns from traffic flowing out (i.e., result sets), the data points act as individual controls, with the top and bottom data points computing and displaying average traffic volume separately. As traffic volume exceeds the norm for the current trend type setting, the data point will gradiate towards red. As with all controls, each point also supports a Windows fly-over that displays detailed statistics and the time of collection.

Since different types of network traffic can be indicative of different problem areas, we will maintain separate "tubes" for each of the major types of network traffic (i.e., user connections, RPC server-server connections, CIS connections for distributed queries, and data movement via Replication Server LTMs and DSIs).

Drill Down Analysis

We have discussed the importance of the diagnostics concept in making it simpler and faster to identify problems. We all know, however, that just identifying them doesn't get you that big raise in salary. Pinpointing the specific root cause and solution, whether its configuration changes or application changes, is critical to your success. To support this need, we offer drill down capabilities dependent on the system component you are looking at. For example, when looking at memory consumption for data caches, it is critical to see which objects (i.e., tables and indices) are frequently requested, so that you can assess which objects should be moved into their own caches. Equally important is deciding how much memory to allocate for those caches. To support this, one can drill down within a single cache to get a list of all its objects and the amount of memory each is currently utilizing.

The list control is a grid that allows the administrator the ability to sort, view, and print large amounts of data. Lists are

modeless dialog boxes to allow multiple copies of the same or different lists to be open at the same time. A subset of the more interesting detailed lists include:

- A *Wait Event List* to show high level bottlenecks through wait events and wait times;
- A *Process Hog List* to identify users that are consuming system resources, including cumulative performance metrics such as CPU time, disk I/O, network traffic, and memory;
- A *Process Wait Event List* to drill into the specific events a given user has been blocked on;
- A *Lock List* to identify processes and objects causing lock contention (locks can be viewed from either a user or object perspective);
- A *Deadlock List* to identify the users, SQL statements, and objects involved in a deadlock;
- A *Named Cache List* to see metrics associated with all named caches such as hit ratios, spinlock contention, and other advanced buffer management metrics;
- A *Cached Object List* to see objects currently cached and the amount of cache they each utilize;
- A *Cached Procedure List* to see procedures currently cached and the amount of cache they utilize;
- An *I/O Activity List* to provide a single view showing I/O across OS devices, Sybase devices, and Sybase Segments;
- A *Segment Detail List* to show storage utilization including growth metrics to estimate "time to failure" as well as performance metrics such as read/write rate information;
- An *Object Activity List* to provide performance metrics such as read/write rates, hit ratios, and access rates at an object level.

From the user level lists, you can also drill into the SQL statement or stored procedure and its graphical query plan along with performance metrics like CPU, disk I/O, and network traffic to allow you to analyze the application's SQL code during normal production hours. Since we are still completing development on this part of the tool, I will defer discussion to a subsequent white paper I will complete by summer.

Architecture and Benefits

The architecture and design of this tool has kept several things in mind. First, many of you work in environments with hundreds, even thousands, of servers. Consequently, DBAs are often responsible for managing multiple servers simultaneously. To support this with minimal administration overhead itself, this tool is designed as a three-tier system with the following high-level architecture.

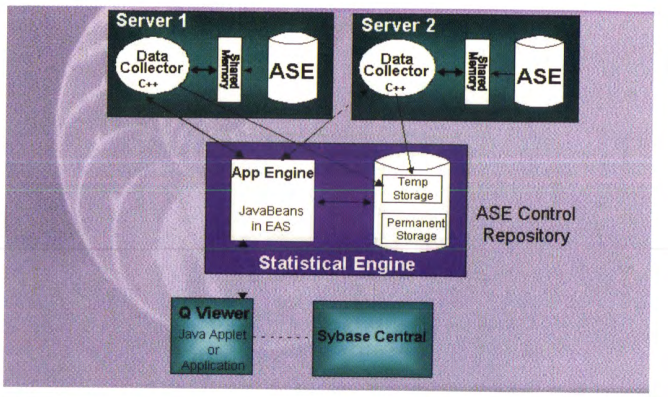


Figure 5

On the host database machine, a lightweight data collection agent talks directly into the ASE's shared memory region to unobtrusively obtain raw performance data. The routine metrics are collected and stored within the Statistical Engine's Repository. This architecture not only provides for simultaneous multi-server diagnostics, but also the statistical trend analysis to provide time-dependent views of the "normal" performance ranges discussed earlier. In subsequent releases (post-Avatar), the repository will also provide us with great extensibility in building historical SQL Statement facilities and proactive tuning and recommendation facilities. Lastly, since we heard from many of you that monitoring and managing systems needed to be as ubiquitous as possible, we have built the Viewer as both an applet, for use within browsers, as well as a stand-alone application.

Conclusion

Performance diagnostics represents a major leap forward in operational manageability. The simplicity and speed with which one can proactively recognize real problems and drill down to correlate it to the root cause, whether configuration or application related, will substantially help DBAs meet their service levels requirements. I only hope you find this tool as enjoyable to use as I have in helping design it!

Lastly, a reminder! If you have a burning question that no one can answer, e-mail me at peter.thawley@sybase.com and include the phrase "Server Views Challenge" at the beginning of subject line. I'll pick those questions that are both challenging and applicable to the audience at large. And remember, when all else fails, blame the network! It'll take them days to figure it out … ■

Peter Thawley is an architect director in the Enterprise Systems Group for Sybase.

PowerDynamo and PowerBuilder: A Dynamic Duo

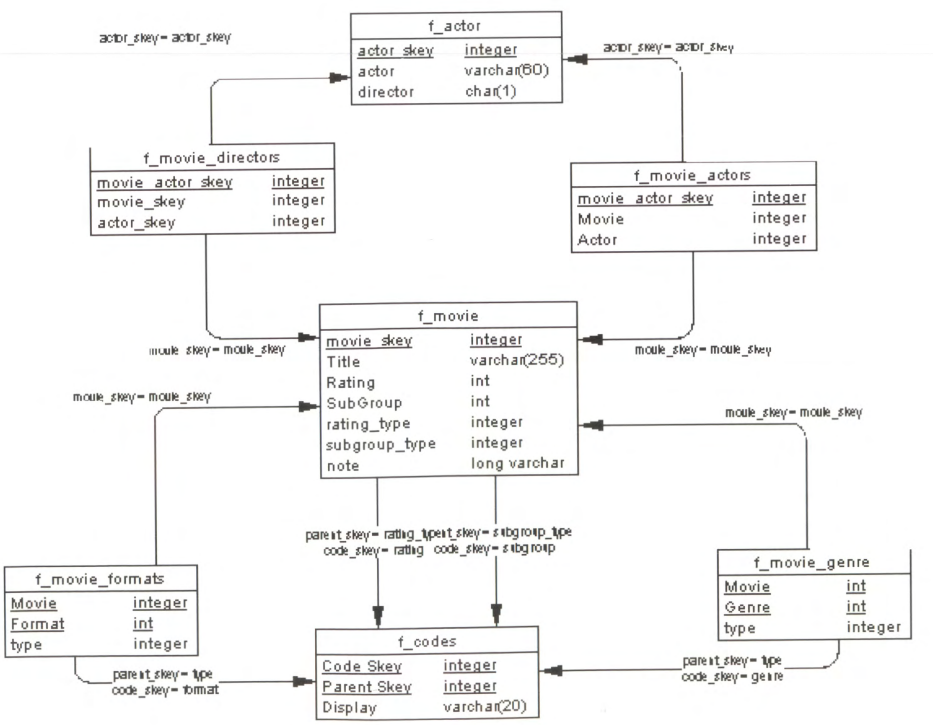
By Thomas Lamb

After you've built that killer PowerBuilder application for the Web, use PowerDynamo to deliver detailed browse results from the same data for the Web. PowerDynamo provides a rich language, based on JavaScript, that allows you to retrieve data from your database and format the output as HTML. During last year's World Cup PowerDynamo was used to provide and deliver web content for the events. This site, supported by two servers, has documented a hit count higher than any previously measured sporting event.

Below, you will find a data model from PowerDesigner (including the datatypes) that provides support for a personal Movie Database built with Adaptive Server Anywhere. After all, even programmers have to have a hobby.

Let's examine a simple Dynamo script that provides a listing of the movies in our database:

```
<HTML>
<TITLE>Simple List of Movies</TITLE>
<BODY>
<H1></H1>
<!--SQL
select f_movie.title
from f_movie
order by title
 -->
<UL>
<!--formatting--><LI><!--/formatting--></UL>
</BODY>
</HTML>
```



This concise script is based on PowerDynamo's standard templates, which allow you to quickly create web pages for display standardized content from SQL statements. The templates even provide a SQL Painter similar to the one used for Datawindows in PowerBuilder.

Let's examine what I did. Within my PowerDynamo site, I selected "Add Template" and provided the template with a name and description (optional). Since my PowerDynamo connection is with my Movies Database, I used the default "inherited" connection. Then I typed in the desired SQL statement. (At this point, PowerDynamo gives you the choice to generate HTML or XML. Note that this provides a fantastic tool and opportunity for you to learn more about XML. However, we are focusing on HTML here.)

Proceed to the next page in the wizard, which gives you an opportunity to choose a standard HTML format. You can choose from Tables (with or without headings), Ordered Lists, Unordered Lists, Definitions, Paragraphs, Detail views, or Links. For this example, I selected Unordered Lists.

There are two units of code in the HTML that are unique to PowerDynamo. The first is bounded by the **<!-<!- -SQL - - >** tags. This construct provides the ability to direct insert SQL into document that will be used by subsequent tags. The next is bounded by the **<!- - formatting - - > <!- - /formatting - - >** tags. These tags denote HTML text that will be repeated for each row in the result set. In this case, **<LI>** is inserted before the column value "title" from the "f_movie" table.

Linking to Other Websites

Let's link our earlier example into the Internet Movie Database (IMDB) at www.imdb.com.The IMDB provides a Title-based search engine. This is accessed using the following link format: http://us.imdb.com/Title?xxx, where xxx is the name of the movie. Now we need to get the name of the movie and insert that into an href link on our way through the result set. Fortunately, PowerDynamo provides another tag, **DATA**, that allows us to specifically identify the column we are interested in. Using this tag we make a simple modification to our template:

```
<HTML>
<TITLE>Simple List of Movies</TITLE>
<BODY>
<H1></H1>
<!--SQL
select f_movie.title
from f_movie
order by title
 -->
<UL>
<!--formatting-->
<LI><a href="http://us.imdb.com/Title?<!--data name=title -->">
<!--data name=title --></a>
<!--/formatting-->
</UL>
</BODY>
</HTML>
```

In this example, you will notice that the DATA tag appears twice. The first time it provides the "title" of the movie to IMDB's Title search engine through an HTML href link. The second time it displays the "title" as the text that is visible in the document to represent the link.

Error Checking

Now, I know that as programmers we never introduce bugs into our code, only unrequested features. However, at some point you will probably want to verify that there were no problems with the SQL you were executing. Let's examine some changes we make to the above templates. First, we'll change the SQL to delay the execution of the query. To do this, we add the parameters NO_EXECUTE and provide a NAME for the query.

```
<!--SQL NO_EXECUTE NAME=qMovies
select f_movie.title
from f_movie
order by title
-->
```

Then we'll add some script to execute the query and check for errors.

```
<!--SCRIPT
  qMovies.Execute();
  if (qMovies.GetErrorCode() != 0 ) {
    document.WriteLn( "<P>" + qMovies.GetErrorInfo() + "</P>" )
  }
-->
```

If there are no errors we can then generate the output.

```
<!--SQL_ON_NO_ERROR NAME=qMovies-->
<OL>
<!--FORMATTING NAME=qMovies -->
<LI><a href="http://us.imdb.com/Title?<!--data name=title -->">
<!--data name=title --></a>
<!--/FORMATTING-->
</OL>
<!--/SQL_ON_NO_ERROR-->
```

Nesting Queries For a Larger Picture

Although the tags work wonderfully for more run-of-the-mill queries, they don't work well when it comes to nesting. This is an unfortunate drawback of this version of PowerDynamo. However, don't be dismayed—with a little more work this can be resolved as well. In the previous example I introduced the SCRIPT tag. Now we'll take a more in-depth look at this tag as we display the actors for each movie.

```
<HTML>
<TITLE>Simple List of Movies</TITLE>
<BODY>
<H1>Simple List of Movies</H1>
Start the script.
<!--SCRIPT
```

Create the query for the Movies.

```
qMovies = connection.CreateQuery( "select f_movie.title, " +
  "f_movie.movie_skey " +
  "from f_movie " +
  "order by title" );
```

Check for any errors.

```
if (qMovies.GetErrorCode() != 0 ) {
  document.WriteLn( "<P>" + qMovies.GetErrorInfo() + "</P>" )
} else {
  document.WriteLn( "<UL>" );
```

Loop through the Movies.

```
    while ( qMovies.MoveNext() ) {
      document.WriteLn( "<LI><a
href='http://us.imdb.com/Title?" + qMovies.GetValue( 1 ) + "'>" +
        qMovies.GetValue( 1 ) + "</a></LI>" );
```

Create the query for the Actors using a subquery with the current movie_skey.

```
    qActors = connection.CreateQuery( "select f_actor.actor " +
      "from f_actor " +
      "where f_actor.actor_skey in " +
      "(select f_movie_actors.actor_skey from f_movie_actors
      where movie_skey = " + qMovies.GetValue( 2 )+ ") " +
      "order by actor" );
```

Check this query for any errors.

```
if (qActors.GetErrorCode() != 0 ) {
    document.WriteLn( "<P>" + qActors.GetErrorInfo() + "</P>" )
 } else {
```

If there are any Actors.

```
      if ( qActors.GetRowCount() > 0 ) {
        document.WriteLn( "<UL>" );
```

Loop through them.

```
          while ( qActors.MoveNext() ) {
```

List the actor name with a link into the IMDB.

```
            document.WriteLn( "<LI><a
href='http://us.imdb.com/Name?" + qActors.GetValue( 1 ) + "'>" +
              qActors.GetValue( 1 ) + "</a></LI>" );
          }
          document.WriteLn( "</UL>" );
        }
      }
    }
    document.WriteLn( "</UL>" );
  }
 -->
</BODY>
</HTML>
```

Summary

The PowerBuilder data model and SQL Anywhere Create scripts are available on the ISUG website. Don't forget to drop me a line at thom@isug.com if there are specific topics you would like to see covered in future issues of this column. ■

How to Set ASE Login Passwords (Back) to Blank

By Rob Verschoor

Officially, ASE login passwords are mandatory: You cannot create a login without a password. When following the normal procedures, there is no way to avoid the use of passwords. To create a login with "sp_addlogin" or change a password with "sp_password", you must at least specify a six-character password, or the operation will fail with an error status.

By itself, this is a good thing, as it improves security. However, sometimes it can be necessary to use a login without a password (a.k.a., a "blank" password or a NULL password). This can be achieved by using a back door trick, as will be explained below.

The only exception to the "mandatory password" rule is when a new server has been created. The "sa" login will have a blank password in this situation. Now consider the following real-life scenario. After a server was created, the "sa" password was left blank for some time, but now you finally changed it to some normal password (as is indeed recommended). However, the next day, some important applications appear not to work anymore because they use a hard-coded, blank "sa" password. To fix this, you would like to reset the password back to blank, but because this isn't possible, your only option seems to be to restore a dump of the "master" database.

Fortunately, there is a solution for this situation. In version 10.0 and later, encrypted login passwords are stored in the "password" column of the "master..syslogins" table as a hexadecimal string. The blank login password for the "sa" login in a new server is not stored as blanks or NULL, but also corresponds to an encrypted hexadecimal string. Using this knowledge, you can set a password to blanks as follows:

1. Create a new server on your platform
2. Log in to this new server as "sa" and run the following query:

```
select password
from master..syslogins
where suid = 1
```

3. Using the hexadecimal string returned by this query, manually update the "syslogins.password" column set a blank password for login "some_login" (Warning: this should be done in a transaction!):

```
update master..syslogins
set password = <hex-string-returned-by-above-query>
where name = "some_login"
```

4. Now you can log into the server as "some_login", using a blank password. Note that this will work for any login, not just for "sa".

Because the encrypted string for a blank password is platform-dependant, there is no common value that will work for all possible servers. You probably won't have to go through the above steps yourself, as you can use the stored procedure "sp_blank_password", which already includes the blank-password strings for the major ASE platforms. This procedure can be downloaded from http://www.euronet.nl/~syp_rob/download.html.

Although not supported or guaranteed by Sybase, the encrypted password strings appear to be version-independent, i.e., they work in all ASE versions on the same platform. Remember that this only applies to ASE version 10.0 and later; earlier versions use unencrypted passwords which can be updated in "master..syslogins" directly. Also note that, in version 10.0 and later, non-blank passwords cannot be shorter than six characters, as was possible in version 4.9.x and earlier. ■

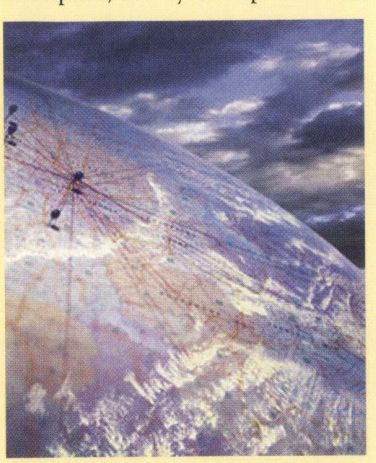
Autometric Spatial Query Server—Spatial Data Management for Sybase Users

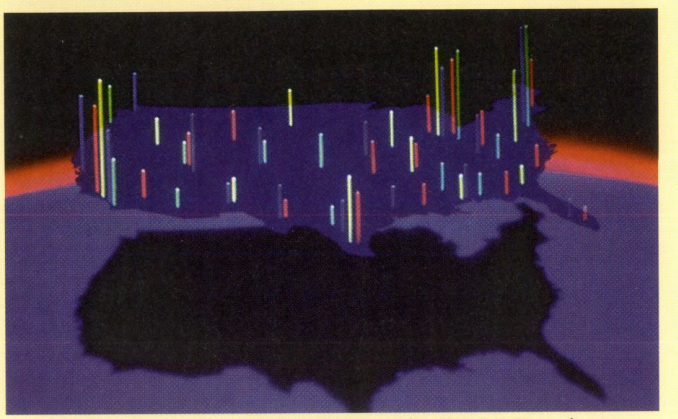
Autometric, Inc., our latest ISUG partner, provides a range of products, services, and solutions for the acquisition, management, analysis, and 3D visualization of digital geographic data. Since its inception as a division of Paramount Pictures in 1957, Autometric has developed information systems for both commercial and government customers.

Autometric's Spatial Query Server (SQS) is a specialty data store for Sybase's Adaptive Server that provides spatial, temporal, and relational data management. It enables its users to add geographic data to their corporate information base, such as points representing customer locations or polygons that represent market areas—allowing them to make business decisions based on geographic factors, such as the number of customers within a market area of interest.

With SQS, spatial data is stored in tables along with traditional relational data types. Whether users are inserting, updating, deleting or retrieving data, SQS treats the spatial data as any other datatype. It also provides users with Spatial SQL, a geographically enhanced SQL that enables more intuitive building of geographic data queries. This interface allows users to more easily build complex queries combining both spatial and relational constraints (e.g., time) in the same query.

Since SQS stores all data in the Adaptive Server Enterprise, the Sybase optimizer decides the best execution plan for complex queries with spatial and relational constraints. Built on Sybase's multi-threaded, Open Server technology, SQS is scaleable, permitting large numbers of simultaneous queries without lag times. Its spatial indexing allows for rapid searches of large data stores.





Autometric's front-end to SQS is the EDGE Developer Option (EDO). In addition to providing a GUI for the acquisition, analysis, and visualization of SQS data, EDO provides access to C++ libraries that allow development of either 2D or 3D whole-earth environment applications. EDO includes classes for inserting imagery, terrain, and map data into the visualization environment. It also takes advantage of the Windows COM architecture to integrate with existing legacy applications. In addition, external data can be added to custom 2D or 3D applications.

Complex simulations involving the spatial relationships between user-defined objects such as satellites, aircraft, ground sites and missiles can be visualized, providing the user with a better sense of the true spatial relationships. In addition, EDO supports the development of applications requiring the fourth dimension, time. This allows the user to visualize a scenario as the object relationship changes over time.

EDO includes an application wizard that allows programmers to quickly create visualization applications for Windows NT, Windows 95, or Windows 98. Users can paste EDO-generated 2D or 3D scenes into PowerPoint presentations or Word documents. In addition, it is possible for users to interact with EDO applications over the Internet via a browser. Data can be retrieved from the Internet and dragged into a custom display application, and EDO applications can also be extended to take advantage of ActiveX.

You can find Autometric at www.autometric.com. ■

Asia-Pacific Conference Draws New Members

The Australasian User Conference, held in Sydney, Australia, on February 22–24 brought ISUG members, Sybase partners, and exhibitors together Down Under for a busy three-day event. Featuring a diverse selection of seminars, sessions, and social gatherings, this annual conference gave the region's rapidly expanding user group a chance to meet and learn about Sybase and Powersoft technologies.

Keynote Speakers and New Technologies

Sybase Chairman John Chen kicked off the conference with a welcome and discussion of Sybase's reorganization into four divisions. Executive Vice President Bob Epstein demonstrated MySupport, Sybase's new online support tool.

New technologies announced at the conference included Sybase's latest version of Sybase Adaptive Server IQ, a line of business intelligence applications, and the new financial server. On the tool side, a considerable amount of interest was generated by the upcoming HTML datawindow. Working in conjunction with the new version of Jaguar, the HTML datawindow offers many advantages when building e-commerce sites and thin-client applications.

Getting the Latest on Upcoming Releases

Users took advantage of a variety of technical sessions as well as pre-conference tutorials. In many cases, the sessions covering new releases were standing room only, as users crowded in to learn about Adaptive Server 12, Enterprise Application Server 3.0 (formerly Jaguar/PowerDynamo), PowerBuilder 7.0, and PowerJ 3.0. The final technical event was the always well-attended enhancements session. ISUG Vice President Thomas Lamb provided an overview of the enhancements process, while Bob Epstein moderated a wide variety of questions from the users.

Opportunities to Network

Users had numerous opportunities to learn from each other throughout the event, with the ISUG booth serving as a central meeting point. This region's rapidly growing user group saw still more expansion as many new members joined ISUG over the course of the conference—giving it a 30% growth rate just in the last six months!

After the intensity of three days of conferencing, attendees wrapped up their stay in Sydney with a Special Event featuring dinner, a stand-up comedian performance, games, dancing, and karaoke. ■

Information in Motion

Sign Up for Sybase TechWave '99

This year, for the first time, the ISUG North American Conference, the Powersoft Conference, the Middleware Technical Summit, and the ISUG European Conference will be combined into one major action-packed event. The culmination of each of these will produce an energized and informative user conference you will not want to miss.

Register to take advantage of over 100 technical sessions on the latest happenings in the areas of Mobile Embedded Computing, Internet Application Development, Data Warehousing, Enterprise Solutions, and Middleware. There will also be extensive training opportunities and hands-on education experiences, as well as a chance to network on common topics in the Special Interest Group meetings. The Exhibit Hall promises to be filled with all the hottest technologies from an extensive list of vendors. And you can't miss the fun at the hospitality suites and Conference Special Event.

Visit the ISUG website at www.isug.com to register today!

NEWS BRIEF

New Internet Applications Releases Available

Sybase Enterprise Application Studio 3.0 (EAStudio) is now available, including Enterprise Application Server 3.0 (EAServer), PowerJ 3.0, andPowerBuilder 7.0. This application package supports eBusiness applications which dynamically create and distribute documents through a secure Internet connection, and allows users to leverage existing expertise to target Internet and Java development. As part of EAStudio 3.0 Sybase introduced the Web DataWindow, which enables developers to create high performance, ultra-thin client Internet applications with a point-and-click functionality.

Sybase Selects New Marketing Vice President

Pamela George has been named as Sybase's new vice president of corporate marketing. She previously served as vice president of corporate communications at Maxager Technology, a decision support software startup company in San Rafael, California. Prior to joining Maxager, she was director of corporate communications at Cisco Systems, and held several positions at Arrow Electronics, Inc., including president of High Tech Ad, Inc., a wholly-owned subsidiary of Arrow, and vice president of corporate communications. She has also worked in marketing at Intel Corp., in sales at Polaroid, and as an editor at Houghton Mifflin.

Sybase's First Quarter Numbers Are Up

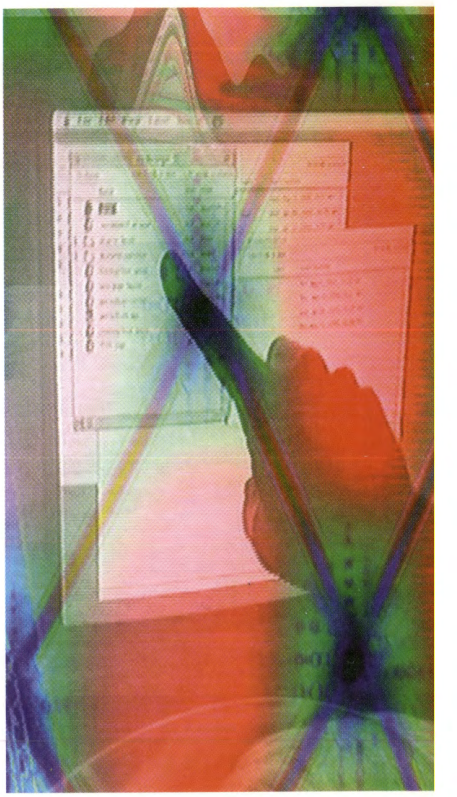
Sybase has announced positive results for the first quarter ended March 31, 1999. Revenues for Q1 1999 were $208.3 million, compared with $206.8 million recorded in the first quarter of 1998. Sybase's first quarter 1999 earnings well exceeded Wall Street expectations with net income of $5.9 million, or $.07 per share, compared to a loss in Q1 1998 of $81.2 million, or $1.01 per share, that included a $51.7 million restructuring charge. Sybase ended the first quarter 1999 with approximately $297 million in cash and investments, up 19% from $249 million in the fourth quarter 1998. Days sales outstanding in accounts receivable for the first quarter were 67 days.

3Com and Sybase to Develop Handheld Database Access

Sybase has teamed up with 3Com to build new wireless enterprise solutions for the Palm Computing platform, including the Palm III, PalmIIIx, PalmV and the upcoming Palm VII. The cornerstone of the solution is Sybase SQL Anywhere Studio's UltraLite and MobiLink technologies, combined with the 3Com CodeWarrior development tool. 3Com demonstrated a prototype at IDG's recent Demo Mobile conference. Sybase's support of wireless technology is expected to present new ways for companies across industries to reduce communications costs, increase employee productivity and better service customers.

Dataquest Names Sybase SQL Anywhere as Market Leader

Dataquest has selected Sybase once again as the mobile database market leader in its 1998 study, "Embedded Database Vendors Focus on Mobile Devices." The report attributes 55 percent market share



to Sybase SQL Anywhere Studio. For three consecutive years, Sybase has been considered the mobile database market leader. The report highlights Sybase SQL Anywhere Studio as "early to market" with technologies key to the mobile arena, including small footprint database and robust replication technologies, as well as support for Windows CE. The mobile database market of $52 million in 1998 is expected to see a compound annual growth rate of 39 percent through 2003. ■

ISUG Focuses on "Year of the Member"

By Linda Morison, ISUG Membership Director

As Sybase continues to reorganize its product divisions and marketing function, ISUG has declared 1999 to be "The Year of the Member." Through the opening months of this year, we have been focusing increased energy on providing better member benefits, more technical information, and supporting your needs and issues during this time of change. We have been working with all four divisions of Sybase so that ISUG can continue to be an advocate for all Sybase, Inc. customers.

In addition, the ISUG board of directors has formed a membership committee that will assist in developing new programs as well as exploring and capitalizing on new user group membership ideas throughout the year.

1999 Membership Benefits

Every year ISUG updates your membership benefits, maintaining favorites such as training discounts, certification exam discounts, the *ISUG Technical Journal*, and a single-user copy of SQL Anywhere Studio. The ISUG website also provides ongoing information to users, with a Members Only section that provides technical information; our streamlined, online product enhancements process; and online member directory. For a complete list of the 1999 membership benefits, please visit us at www.isug.com.

By the way, be sure to bookmark our site and keep checking back, as it is refreshed every five to six weeks. If you're interested in starting your own local user group, we've just started up a new section containing information on how to get one going. Also, take a few minutes to verify your contact information in the Members Only section of the website. Keeping this current is the only way to insure that you will get the most out of your membership.

You gain access to the ISUG website's Members Only section with a valid user id and password, which new members will receive by e-mail. Renewal members retain the user id and password from the previous year. If you have forgotten your password, please visit the site to request a new one.

ISUG is now quickly processing membership applications and renewals, and distributing the 1999 Welcome Packet. If it has been more than eight weeks since you submitted your application and you have not received your Welcome Packet, please contact us at info@isug.com.

Share the Knowledge—Bring a Friend

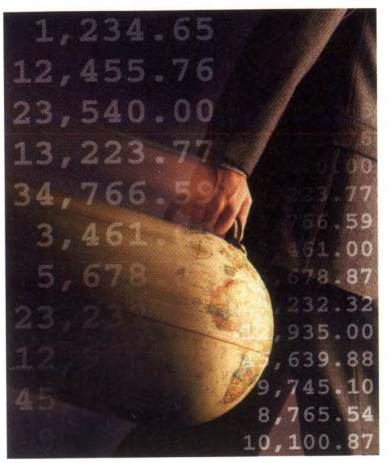
I am always surprised when I meet a Sybase customer who is not a member of ISUG, yet I know there are still many of them out there. To continue our mission of acting as an information channel, an advocate for our members, and a force that influences Sybase product direction, we are focusing on strengthening the ISUG organization.

As part of our "Share the Knowledge–Bring a Friend" initiative, we are asking members to tell a friend about ISUG. Remember that ISUG is not just for DBAs. Users of all Sybase products—development tools, middleware, and database—benefit from ISUG membership. The ISUG website is an excellent tool for explaining the benefits. I have given many colleagues personal tours of the site, including the Members Only section, and they always find the online *Journal* articles and enhancement process very interesting.

Remember that as an extra bonus, ISUG members attending the Sybase TechWave '99 conference will receive a $75 discount, making membership free to attendees. ■

Calendar of Events

User Group Meetings and Events



North American and Powersoft User Conference
Orlando, FL
August 22-26, 1999

LUG Meetings

Chicago, IL
LUG Meeting: May 20

Houston, TX
LUG Meeting: June 17

Indianapolis, IN
LUG Meeting:
Second Tuesday of even months

Northeast (SNEKUS)
LUG Meetings: June 2

Oregon
LUG Meetings:
First Tuesday of even months

Pacific Northwest (Seattle)
LUG Meetings:
First Wednesday of each month

Rocky Mountain
LUG Meetings:
Second Tuesday of even months

Southern Virginia
LUG Meetings:
Third Tuesday every other month

INTERNATIONAL

Sybase User Group

Let us know what your user group is doing! Contact Sybase User Group Marketing with the dates and locations of your group's meetings and special events. They can be reached at: Phone 510-922-7525, Fax 510-922-0882 or Email isug@sybase.com

INTERNATIONAL

Sybase User Group

1999 ISUG Membership Application

ISUG Benefits

- A subscription to the quarterly ISUG Technical Journal
- Adaptive Server Enterprise 11.9.2 New Functionality book
- Voucher for Sybase Adaptive Server Anywhere Studio
- Access to online ISUG enhancements voting and reports
- Access to online ISUG Membership Directory
- Access to Special Interest Group and Enhancements mailing lists

Education Discounts

- Save 10% on all Sybase education classes
- Save 20% on Sybase training products and programs
- Save 20% on Sybase Certification exams
- Save $75 off on registration for ISUG-affiliated conferences

please check one:

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

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

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.

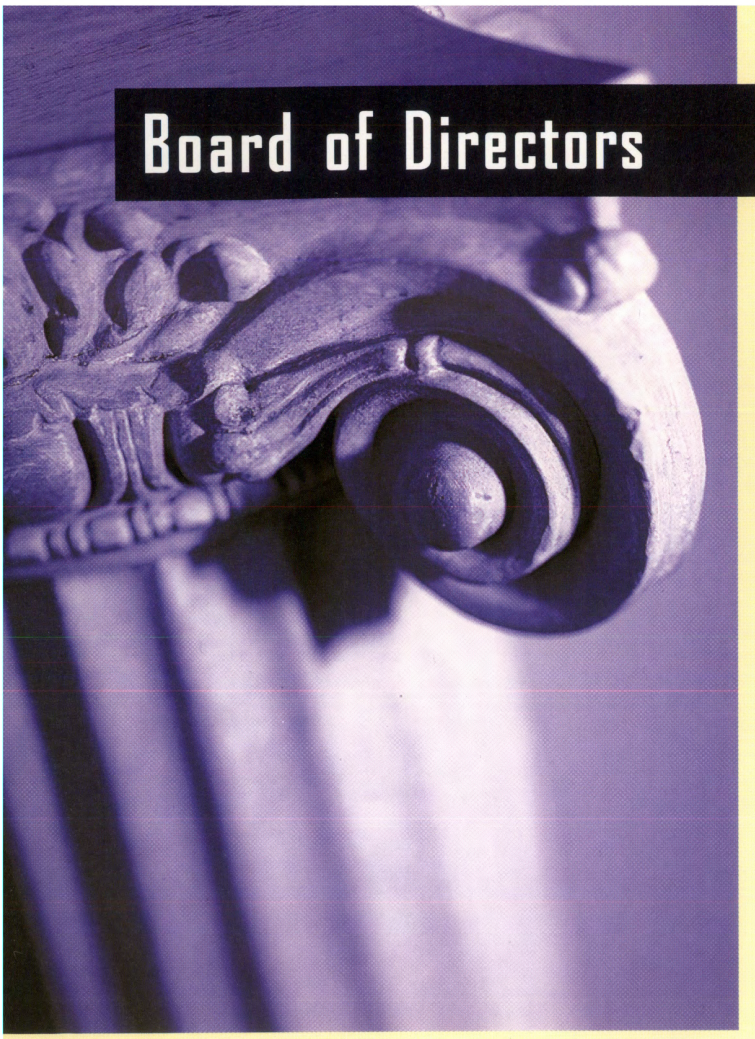
signature _____ date _____

SOURCE CODE: S S J Q 2

Mail form to...

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

Board of Directors



President

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

Vice President

Thomas J. Lamb
PowerCerv
Phone: 847-397-1572
t.lamb@bigfoot.com

Secretary

Karen Pursch
Sybase
Emeryville, CA
Phone: 303-604-2545
karenk@sybase.com

Treasurer

Kathy Ridley
Amoco
Houston, TX
Phone: 281-366-2334
kathy_a_ridley@amoco.com

Conference Director

Cynthia Gill
DRT Systems
Houston, TX
Phone: 713-868-5537 x109
Fax: 713-868-4014
cynthia_gill@notes.drthou.com

ISUG Technical Journal Director

Teresa A. Larson
Group I Software
Phone: 301-918-0896
Teresa_Larson@g1.com

Enhancements Co-Directors

Jibu Abraham
Ajilon Software
Phone: 303-221-5428
Jibu_abraham@hotmail.com

Jaideep Chawla
Pricewaterhouse Coopers
Phone: 703-645-5882 x8447
Jaideep.Chawla@us.pwcglobal.com

Membership Director

Linda Morison
Automated Data Sciences/CADscan
Phone: 202-205-6717
Fax: 202-205-7064
lmorison@erols.com

Australasia RUG Director

Peter Brouwer
MMI Insurance Group
Sydney, Australia
Phone: +612 9390 6222
Peter_Brouwer@notes.mmi.com.au

Asia RUG Director

Joseph Fong
City University of Hong Kong
Hong Kong
csjfong@cityu.edu.hk

Electronic Media Director

Michael Peppler
mpeppler@mbay.net

Members-at-Large

William Niemi
Fidelity Investments
Boston, MA
Phone: 617-563-8787
Fax: 617-476-6282
Bill.Niemi@fmr.com

Daniel Kenyon
Vantive Corporation
Oakland, CA
Phone: 510-763-1821
Fax: 510-208-5660
Daniel@vantive.com

Partner Membership Director

Frank Monteverdi
DRT Systems
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.
Phone: 281-890-3196
Fax: 713-918-1173
cynthia_bean@bmc.com

SIG Co-Directors

Robert M. Craig
Phone: 713-868-5537, x472
bob_craig@drthou.com

Jay Hunt
djayhunt@mvp.net

Sybase Contact:

Yared Benyam
User Group Marketing
Emeryville, CA
Phone: 510-922-7525
Fax: 510-922-0882
isug@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

Finland
Kristina Salminen
Sanoma Corporation
Phone: 358 9 122 87 3184

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

Germany
Jens Timmermann
TOPOLOGIX GmbH
Phone: 49-40-352-253
Fax: 49-40-340-340
jens@suntana.topologix.de

Ireland
Tracy Egan
Sybase Products Ltd.
Phone: 01 6776777
Fax: 01 6776614
tracy@sybase.ie

The Netherlands
Jolanda Zwaan
Ra.zwaan@tref.nl

Norway
Jan Kare Fjeldstad
Aker Elektro
Phone: 47-53-49-23-55
jkf@ael.aker.no

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

Sweden
Ulf Warmlov
SuperTech AB
Phone: 46-8622-6227
Fax: 46-8768-7862
ulf.warmlov@supertech.se

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

Turkey
Levent Sensezgin
Info Sybase Yazilim A.S.
Phone: 90-212-275-8995
Fax: 90-212-273-0612
lsensezgin@info.com.tr

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

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

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

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

Northeast "SNEKUS"
Bruce Driver
Consultant
Phone: 508-443-3310
bdriver@compuserve.com
www.snekus.com

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

Ottawa
Tony Antonallo
Phone: 613-798-7507

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

Southeast Region

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

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

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

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

Tampa Bay Area
Karl Althaus
Kt93@hotmail.com

North Central Region

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

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

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

Indianapolis, IN
Jim Strange
Anthem IT
Phone: 317-488-6264
Fax: 317-542-6550
jim_strange@aici.com
www.cs.bsu.edu/homepages/sam/isug

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

Omaha, NE
Sybase, Inc.
Phone: 303-486-7700

Winnipeg
Jim Novisat
Phone: 204-946-7748
Fax: 204-946-4567
jzn@gwl.ca

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

Michelle Murphy
Wisconsin Electric Power Co.
Phone: 414-221-2068
Fax: 414-221-4744
michelle.murphy@wisenergy.com

SW-Mountain Region

Regional Contact
Cynthia Gill
DRT Systems
Phone: 713-868-5537
Fax: 713-868-4014
cynthia_gill@notes.drthou.com

Austin
Rob Gustavson
BCS Systems, Inc.
Phone: 512-921-0074
rgustav@swbell.net

Dallas
David Straiton
Computer Language Research, Inc.
Phone: 972-250-7989
Fax: 972-447-6082
david_straiton@clr.com

Rocky Mtn. Denver
Matt Townsend
Ayupp, Inc.
Phone: 719-488-2314
matt_townsend@earthlink.net

Houston (SUG-TX)
Robert M. Craig
DRT Systems
Phone: 713-868-5537 x472
Fax: 713-868-4014
bob_craig@drthou.com
www.sugtx.org

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

Kansas City (KCASUG)
George Meiers
Hallmark Cards, Inc.
Phone: 816-545-6395
Gmeier1@hallmark.com

Far West Region

Regional Contact
Matt Webber
Arizona State University
Phone: 602-965-2186
mwebber@asu.edu

Arizona
Reny Abrego
Phone: 602-402-6491
reninaz@aol.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

Northwest Region

Regional Contact
Marian Abrams Pasela
Vantive Corporation
Phone: 530-757-4455
Fax: 530-753-9291

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

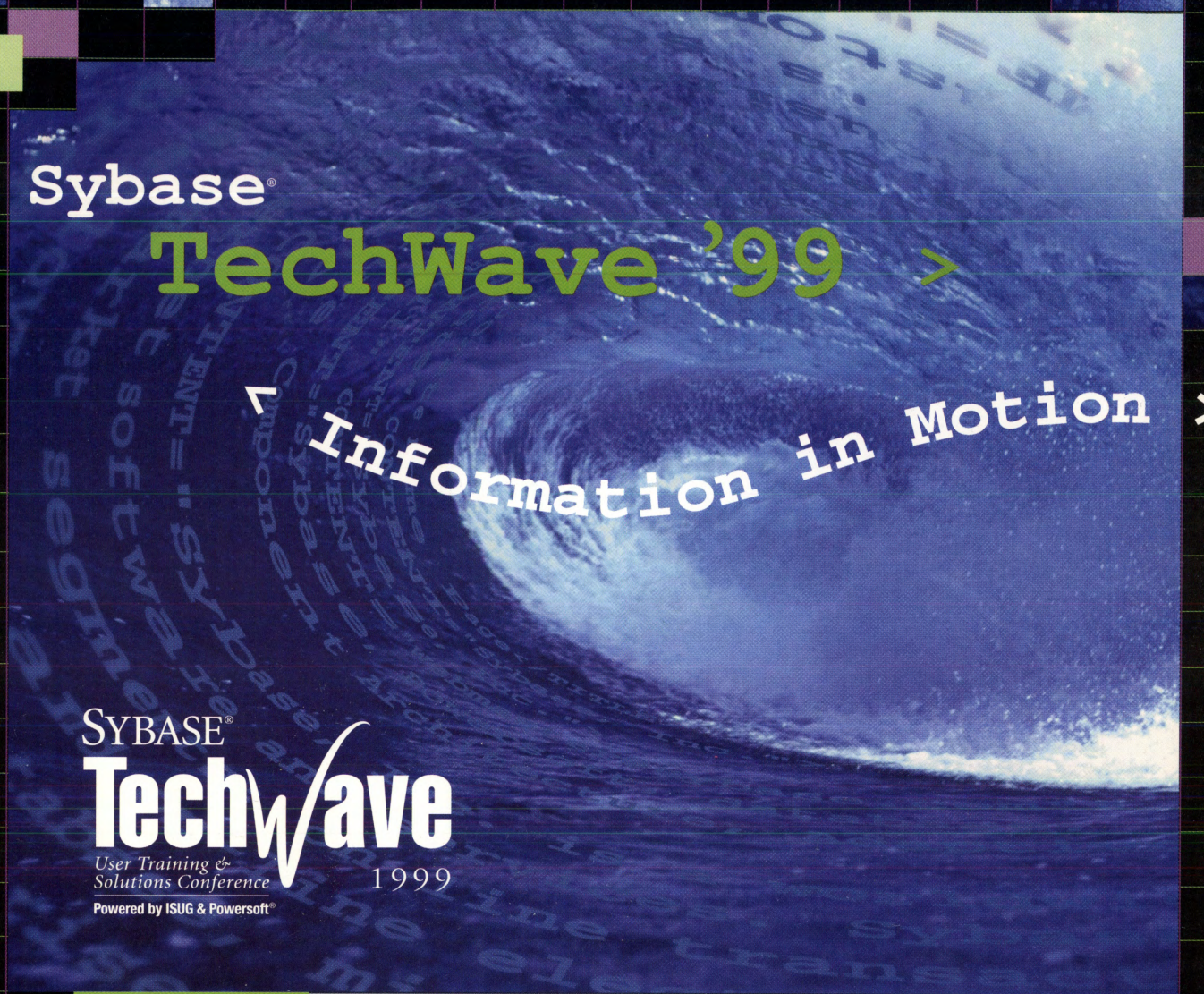
Calgary/Edmonton
Khristine Harty
Visual Systems, Inc.
Phone: 403-262-9970 ext. 1
Fax: 403-262-9966
kharty@visual.com

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

Sacramento
Peggy Turner
Vantive Corporation
Phone: 530-750-7400
Fax: 530-750-9291
peg@vantive.com

San Francisco Bay Area "SUGBAY"
Peggy Turner
Vantive Corporation
Phone: 530-750-7400
Fax: 530-750-9291
peg@vantive.com

For information on PowerBuilder Local User Groups, please contact Effie Panagiotakis at 978-287-2687 (phone) or epanagio@sybase.com.



< Register early and save! >

Register on or before Friday, May 21, 1999, pay only **$995**

Register between May 22 and June 25, 1999 and pay **$1,095**

Register after Friday, June 25, 1999 and pay **$1,195**

A User Training and Solutions Event

At Sybase, we are committed to giving our users new opportunities to learn about our products and our company. So for the first time, we're combining our annual International Sybase User Group (ISUG) North American Conference, the Powersoft Conference, and the Middleware Technical Summit into one unified event—Sybase TechWave '99.

Creating one comprehensive conference means even more in-depth product training, more one-on-one time with experts, and more chances to experience the full range of Sybase's product offerings. Everything you need to put Information in Motion!

Join us for Sybase TechWave '99 and experience the most comprehensive user event in Sybase history. For registration information and event updates, visit us at **www.sybase.com/techwave99** or call **(508) 652-1008.**

< August 22-26,1999
The Walt Disney World Swan and Dolphin
Lake Buena Vista, Florida, USA >

SYBASE®
INFORMATION ANYWHERE™

sybase.com/techwave99